Creating Automation to Couple Laser Light from an Optical Fiber to a Photonic Integrated Circuit

Undergraduate Honors Thesis

Presented in Partial Fulfillment of the Requirements for Graduation with Honors Research Distinction in Electrical and Computer Engineering at The Ohio State University

By

Aaron P. Maharry

May 2017

Advisor

Professor Ronald M. Reano

The Ohio State University

Department of Electrical and Computer Engineering

Abstract

Coupling light between fiber optic cables and photonic devices on silicon chips is important for optical communications. Recently, vertically bent cantilever couplers have been developed that have better bandwidth and lower power loss than grating couplers, which are the current standard for fiber-to-chip coupling. A system that can perform butt-coupling between optical fibers and cantilever couplers on an integrated silicon photonic device is reported here. This system is able to perform butt coupling using vision processing and measured coupled power as feedback, and requires minimal human assistance. This system can be extended to automatically collect measurements from multiple devices on the same wafer, enabling large scale testing of devices with cantilever couplers in research and manufacturing.

Acknowledgements

I would like to thank my advisor, Professor Ronald M. Reano. His advice helpful discussions were instrumental throughout the progression of this project. I would also like to thank Dr. Michael Wood for assistance in the laboratory and for insightful discussions about the challenges associated with this project. I would also like to thank Tyler Nagy for assistance in modifying the laboratory camera mounting setup. Additionally, I would like to thank Ryan Patton and Eric Martin for their helpful discussions and ideas. Finally I would like to thank Professor George Valco for sitting on my thesis defense committee.

This research was supported by the College of Engineering Undergraduate Research Scholarship.

Table of Contents

1. Introduction

1.1. Background

Integrated optical devices are essential components in optical communications systems. Silicon photonics has the potential to allow large scale integration of many photonic devices on a single chip while using the well-developed complimentary metal-oxide-semiconductor (CMOS) fabrication processes prevalent in the electronics industry. This allows for low cost manufacturing of photonic devices, and for integration of photonic and electronic devices on the same chip. In order to take advantage of integrated optics for high bandwidth optical communications processing, devices must be designed to couple light from a fiber optic cable to an integrated waveguide, and vice versa.

Currently, grating couplers are the standard couplers used in industry and research. They have the advantage of being able to couple light from anywhere on the surface of the chip, as opposed to earlier designs, which were limited to coupling at the edges of a diced chip [1]. Grating couplers, however, suffer from reduced power coupling efficiency and have a limited bandwidth due to their periodic structures. Recently, vertically bent inverse width taper cantilever couplers have been demonstrated at the Integrated Optics Laboratory at The Ohio State University [2-3]. A diagram of the demonstrated device is shown in Figure 1. The cantilever coupler is fabricated on a silicon-on-insulator (SOI) wafer, and consists of a silicon substrate, an SiO_2 buried oxide (BOX) layer, an etched silicon waveguide, and an SiO_2 oxide cladding deposited by plasma enhanced chemical vapor deposition (PECVD) [3]. These layers are shown in Figure 1. A tapered optical fiber is then brought into contact with the cantilever structure to form a full coupler. Since the waveguide is bent out of the plane of the chip with thin film stress these cantilever couplers can couple light anywhere on the surface of the chip to an optical fiber. In addition, since inverse width

tapers are used instead of periodic gratings, cantilever couplers have superior performance compared to conventional grating couplers [2]. Figure 2 shows how light couples to the tapered optical fiber through an inverse width taper.

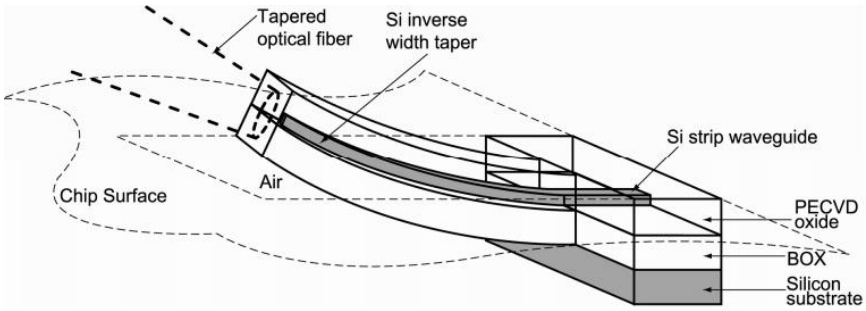

Figure 1: Diagram of a vertically bent cantilever coupler [2].

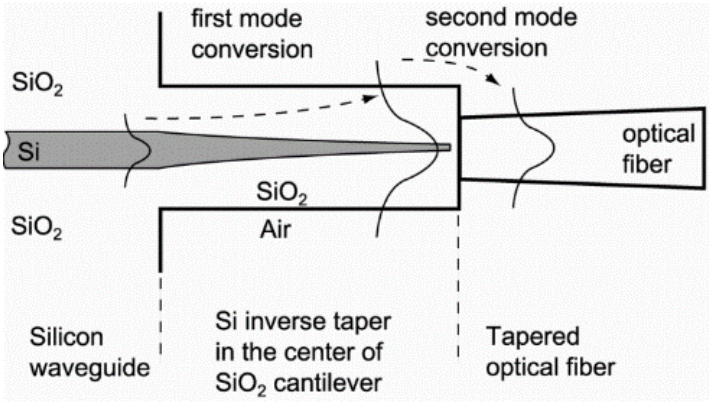

Figure 2: Light coupling from an inverse taper waveguide coupler to an optical fiber [2].

With current technology, grating couplers have around a 1.5 dB power coupling loss [4-5]. Due to their periodic structure, their coupling efficiency drops significantly with small changes in the wavelength of the light being coupled. For example, bandwidths of around 54 nm for a wavelength dependent coupling loss less than 3 dB have been reported [1, 5]. In comparison, power coupling losses of 0.5 dB and 0.62 dB for the TM (transverse magnetic) and TE (transverse

electric) modes, respectively, have been demonstrated in cantilever couplers [3]. Cantilever couplers also have less than 1 dB total power loss over a 103 nm bandwidth, a significant improvement over grating couplers [3]. The cantilever couplers reported in [3] are the state of the art for fiber-to-chip coupling in silicon photonics.

Vertically bent cantilever couplers rely on manual alignment for coupling between the waveguide and optical fiber [2-3, 6-7]. Manual alignment is performed by moving the optical fiber with 6-axis positioning stages, and alignment feedback is provided by real time microscope images and power transmission measurements. Butt-coupled cantilever couplers, unlike grating couplers, require the fiber to touch the cantilever, and thus are sensitive to small coupling misalignments. In [7], a lateral misalignment of just 0.5 μm was enough to lower the power coupling efficiency by 1 dB. In order to achieve the required level of accuracy, a researcher typically spends 30 minutes aligning the input and output fibers to test each photonic device. Silicon wafers that contain hundreds of devices are common, and require many hours of manual alignment to test every device. In the process, human error can manifest itself in misalignment leading to unreliable measurements, and even broken optical fibers.

There is a need for an automated system to perform butt-coupling and testing of photonic devices with cantilever couplers. Similar systems for grating couplers exist in industry [1, 8], but are not capable of performing butt-coupling, which is required for high performance cantilever couplers. This system would decrease the time spent manually coupling and testing devices, and improve coupling alignment and measurement reliability. Additionally, this system would validate cantilever couplers' compatibility with wafer scale manufacturing testing.

1.2. Significance

State of the art cantilever couplers have better coupling loss and bandwidth characteristics than conventional grating couplers, but the lack of automated butt-coupling and testing systems is a barrier to large scale adoption of the technology. The system presented can perform the basic coupling operations required for a more complete automatic testing platform that uses cantilever couplers. The current system can also be used as a human aid during the butt-coupling process. Once extended to perform automated chip-scale testing, the system would enable the usage of high performance cantilever couplers in device characterization and large scale manufacturing testing.

1.3. Literature Review

Similar wafer-scale automated testing systems have been demonstrated using grating couplers [1, 8]. These systems differ from the system reported here in that cantilever couplers require the optical fiber to be butt coupled to the device, whereas grating couplers do not. The need for butt-coupling introduces additional design challenges that haven't been addressed in the literature. One such challenge is the need to automatically detect when the optical fiber has contacted the coupler on the device. Another unaddressed facet of butt coupling that is not treated in the literature is that during alignment, before coupling has occurred, the fiber position is subject to semi-random vibrations, and the position of the fiber must be described in terms of a statistical distribution that collapses to a single point once butt-coupling is achieved and the fiber is held in place. This adds additional requirements to an automated coupling system in order to achieve desired levels of reproducibility in the final coupling loss.

2. Design

2.1. Experimental Setup

The automated coupling system is composed of fiber positioning stages, optical and infrared cameras, an optical power meter, a tunable laser source, and a controlling computer. These components are described below. The full system is shown in Figure 3.

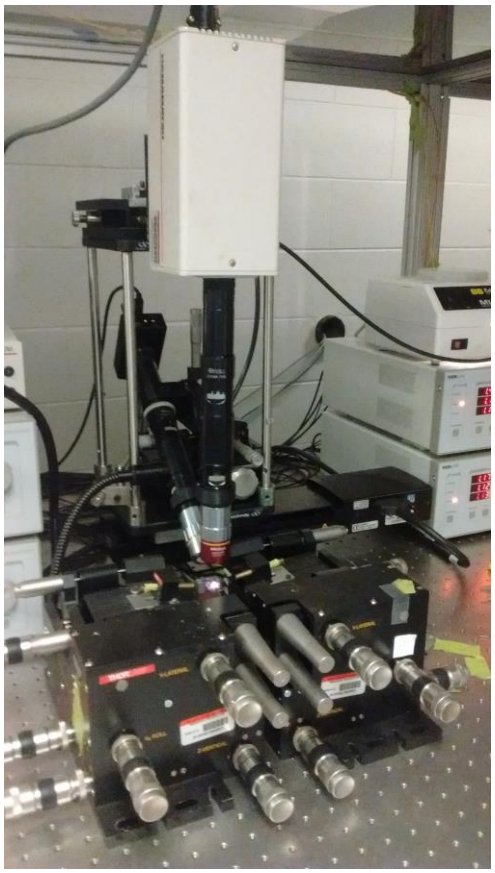

Figure 3: Experimental setup of the automated coupling system.

2.1.1. Cameras

Two microscope cameras are used to perform coupling. The first camera is mounted vertically, and views the device being coupled to from above. This camera captures infrared light,

and records a grayscale image with a resolution of 640×480 pixels. The second camera views the device being coupled to from the side, at an angle of approximately 45° above the horizontal. This camera captures visual light, and records a grayscale image with a resolution of 1392×1040 pixels. Both cameras have manually adjustable zoom and focus. These are set manually before coupling is performed. The zoom for the side-mounted camera is set to the maximum value 7x, and the zoom for the top-mounted camera is set to 3.5x. The relationship between pixel size and real distances will be discussed in section 2.2.1 of this thesis. The accompanying illuminators for each camera are also set manually prior to performing coupling.

The existing hardware design was modified so that both of the microscope cameras are mounted on a 1-axis motorized stage (Newport ILS150CCHA), which enables automatic control of lateral camera position. The stage has a 150 mm travel, and a resolution of 0.1 μm. Automatic control of the stage is facilitated by a motion controller (Newport ESP301) that communicates with the computer via a serial interface.

Camera movement is required for device scale coupling in which both the input and output fibers need to couple to the device concurrently. The system was designed to support testing of devices that are arbitrarily long, but both of the cameras have limited fields of view. Therefore, the motorized linear stage must be used to translate the cameras between each device's input and output couplers to enable a full automatic two-port coupling process. Both cameras are mounted on the same stage, so their motion is synchronized.

2.1.2. Fiber Positioning Stages

Two 6-axis fiber positioning stages (Thorlabs MAX603D) are used to adjust the positions of both the input and output optical fibers for coupling. The stages have manual controls for coarse alignment, as well as piezoelectric actuators that are used for automatic coupling. The piezoelectric

stages have a travel range of 30 μm in each of the three linear axes, and 0.52 mrad of range in each of the angular axes. The piezoelectric stages have closed loop control with a resolution of approximately 100 nm.

2.1.3. Laser

A tunable laser is used to inject light into the coupling system. Light at a wavelength of 1550 nm and a power of -10 dBm is injected into the system to aid with power measurement based alignment and coupling. Once both fibers have been coupled to the device under test (DUT), the laser can perform wavelength and power sweeps to measure the device performance. The tunable laser is able to be controlled remotely from the computer, enabling automatic performance of test recipes, although this has not been implemented in the current system.

2.1.4. Power meter

An optical power sensor and power meter are connected to the output optical fiber of the system. The power meter can measure the output power of the system, which is used as feedback for fiber alignment for butt coupling, as well as for measuring device transmission loss during device testing. The noise floor of the optical power sensor is -80 dBm, and its resolution is 0.01 dBm.

2.2. Automated Coupling System

The automated coupling system is designed to automatically perform both fiber-to-fiber and fiber-to-chip coupling. Fiber-to-chip coupling is used to couple to actual on-chip photonic devices for testing. On the other hand, fiber-to-fiber coupling, in which two optical fibers are butt-coupled to each other, is primarily useful as a way to measure the baseline optical insertion loss of the system independent of the characteristics of the DUT and its associated couplers. This

measurement can then be used to extract the insertion loss of the DUT and its associated couplers from subsequent measurements.

The software for the automated coupling system is written in MATLAB, and a graphical user interface (GUI) is provided for the user to operate the system. The software is broken into modules for vision processing, actuating the piezoelectric stages, and executing the algorithms for fiber-to-fiber and fiber-to-chip coupling. The GUI, shown in Figure 4, allows the operator to run automatic coupling algorithms and monitor the current state of the system. The mode selection menu in the top-left corner of the GUI allows the user to select from a number of available modes of operation. The real-time camera images for both the side-mounted and top-mounted cameras are also present in the top right corner of the GUI shown in Figure 4. The results of the vision processing, the measured coupled power, and the current positions of the piezoelectric stages are all available for the operator to monitor during automatic coupling. See Appendix A for a more detailed description of the user interface functions.

The GUI also has controls that allow the user to manually jog the piezoelectric stages, as well as the motorized camera stage. This, combined with real-time camera images and coupled power measurements provides the user with the necessary tools to perform manual coupling. In addition, the GUI gives the user the option of displaying the detected locations of the fibers and cantilever couplers on the camera images, which can serve as an alignment aid for a human operator performing coupling.

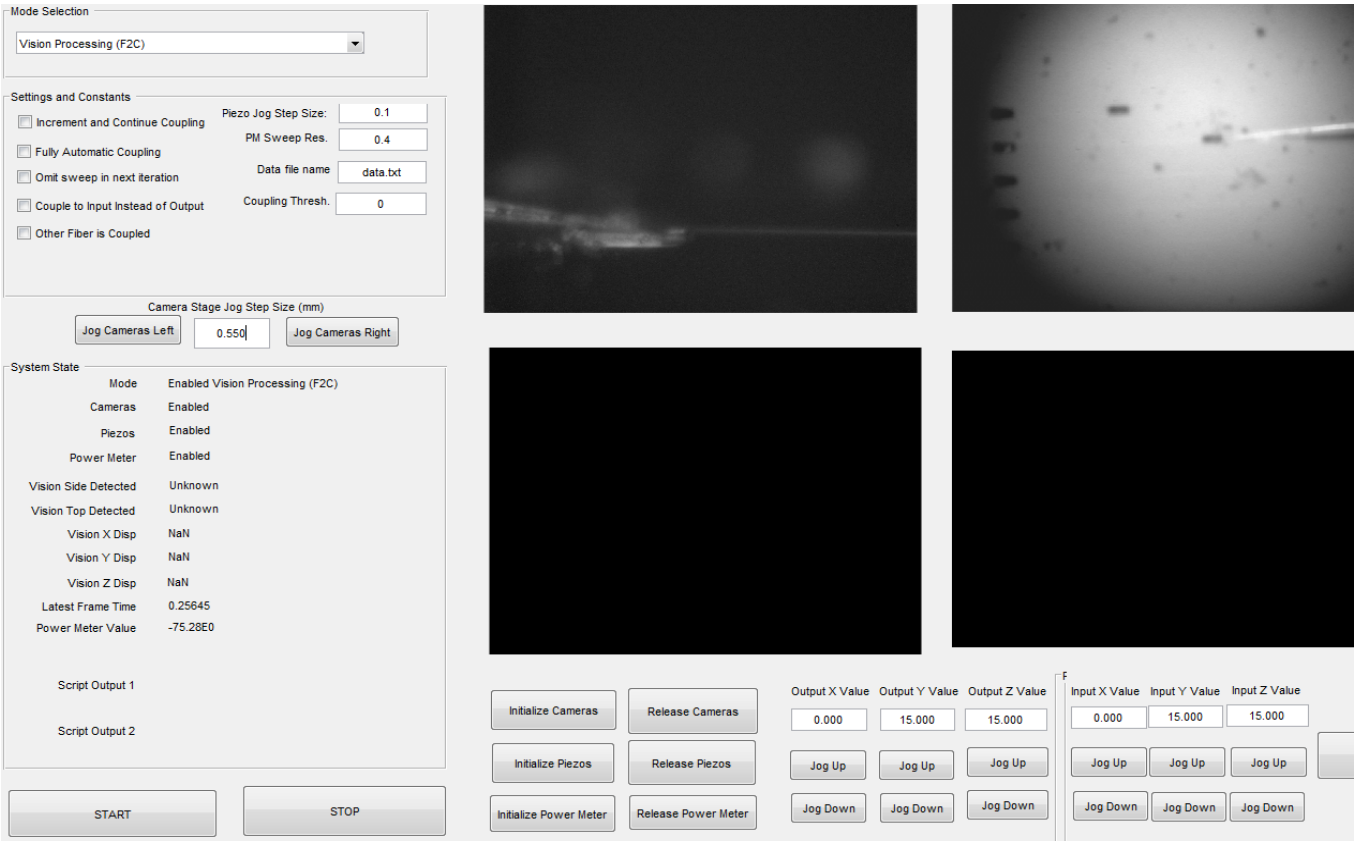

Figure 4: General User Interface (GUI) for the automated coupling system.

2.2.1. Vision Processing

The goal of the vision processing algorithm for the automated coupling system is to estimate the relative displacement in 3 dimensions between the two fibers or between the fiber and the cantilever, depending on if fiber-to-fiber or fiber-to-chip coupling is being performed,

respectively. The image from the side mounted camera is used to measure the axial (*x*-axis) displacement, and the image from the top mounted camera is used to measure the lateral (*y*-axis) displacement. Since the side camera is mounted at an angle of 45° relative to the horizontal, information from both the top and side camera images is combined to estimate the displacement in the vertical (*z*-axis) direction using the following equation:

$$\Delta z = \frac{[\Delta z \; from \; side]}{\cos(45°)} - [\Delta y \; from \; top] \cdot \tan(45°)$$

All of the displacement estimates are scaled by empirically determined constants to convert from pixel distances to real lengths. For the side camera images, the conversion factor is 0.14 μm per pixel, measured at full (7x) zoom. For the top camera images, the conversion factor is 0.75 μm per pixel, measured at 3.5x zoom. Since the camera zoom settings are not changed during automatic coupling, these conversion factors will remain valid throughout the coupling process.

2.2.1.1. Fiber Detection

The first step in the vision processing algorithm for detecting an optical fiber in a side camera image is to apply an adaptive threshold to the grayscale image to select areas of the image that potentially contain a fiber. Next, morphological transformations are applied to filter out noise and smooth out the shape contours. The results are then analyzed to detect all of the connected pixel contours inside of it. These contours are shown in white in Figure 5. The convex hull, enclosed area, aspect ratio, and position of each contour are then computed. The convex hull operation creates a convex contour that fully encloses the original contour, and is useful for eliminating noise and accurately detecting the fiber. The aspect ratio is the ratio between the lengths of the long and short sides of the minimum enclosing rectangle of the contour. Based on these metrics, the largest contours with acceptable aspect ratios and positions on the screen are selected to be fibers. Based on their relative positions in the image, the input fiber can be

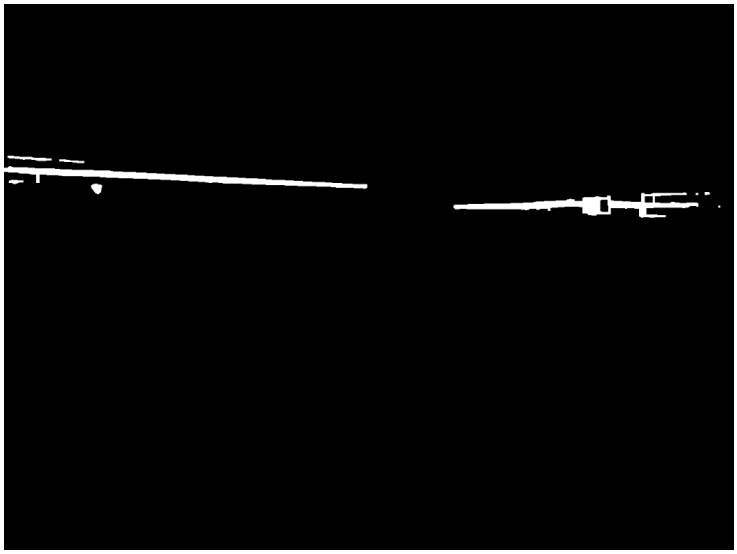

Figure 5: Results of thresholding and morphological transformations on a side camera image.

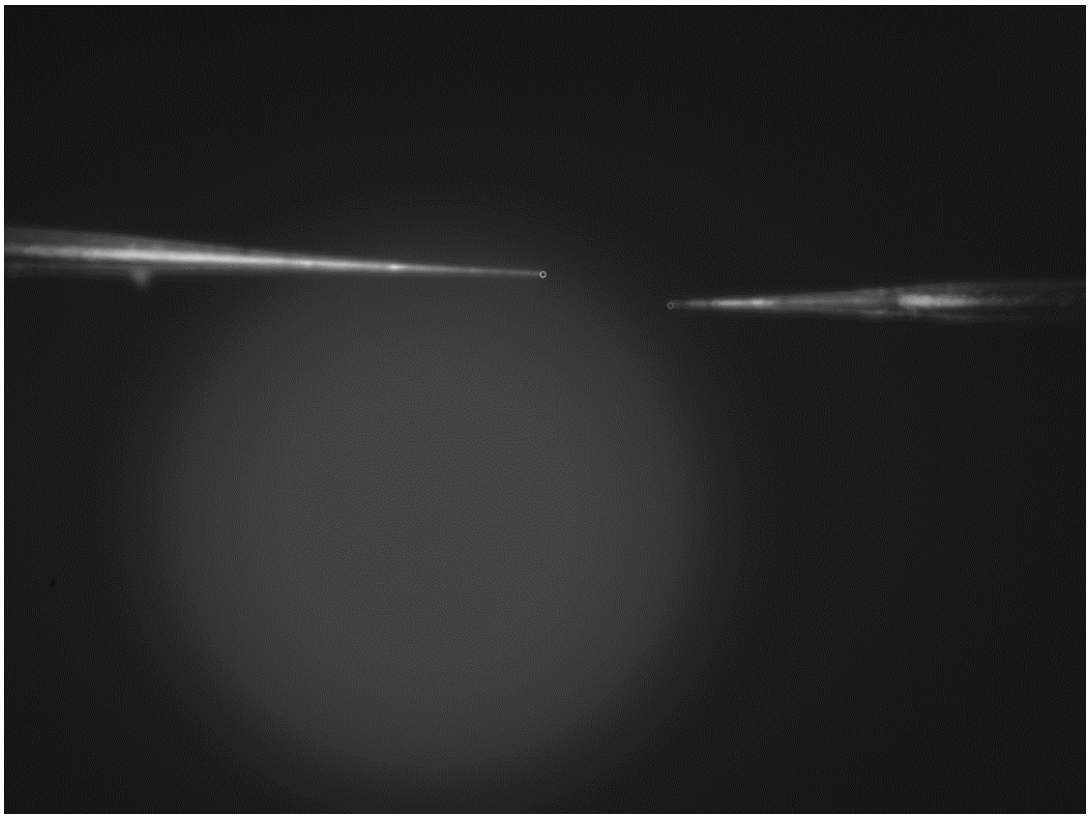

Figure 6: Processed side camera image. Grey circles indicate the estimated fiber tips.

distinguished from the output fiber. This is used in the vision processing algorithm for both the fiber-to-fiber and fiber-to-chip coupling algorithms to identify the correct fiber. Thresholds for

acceptable shape and position were determined empirically. Once the fiber contours are known, the position of the tip of each of the fibers is determined by estimating the position of the extreme end of the fiber contour. The result of this step is shown in Figure 6. Finally, the displacement in microns can be computed as discussed above.

The vision processing algorithm for detecting fibers in a top camera image is similar to that of the side camera processing algorithm. However, due to different illumination conditions, it isn't possible to reliably estimate the exact location of the tip of the fiber. Thus, once the contour of each fiber is detected, a line of best fit for the contour is computed. By extrapolating the line of best fit to the centroid of the opposite contour, the lateral displacement can be computed. Figure 7 shows an image from the top camera with gray lines representing the lines of best fit, and white lines representing the estimated lateral displacement. In a case where two fibers are detected, the two estimates are averaged to produce a final value.

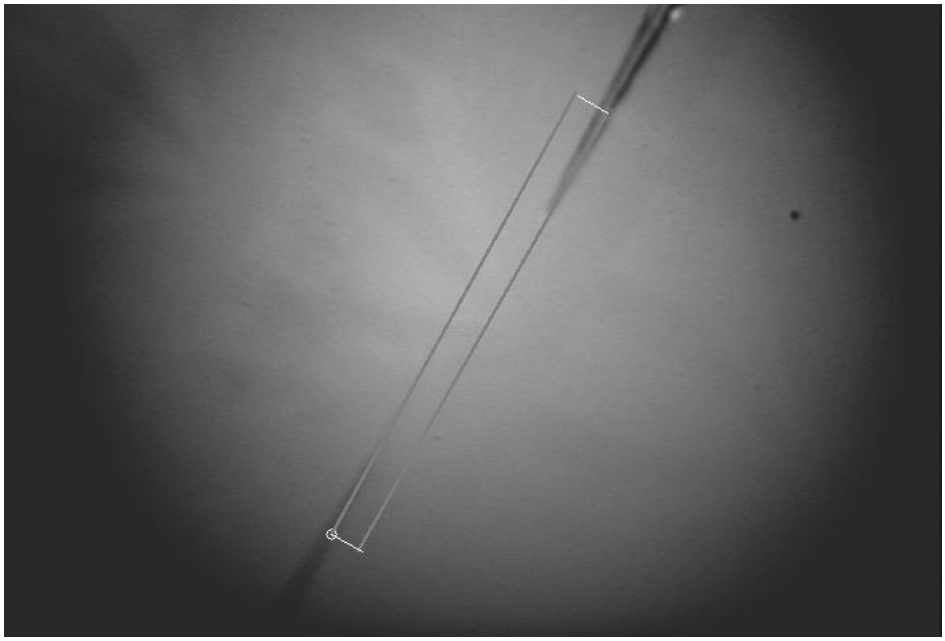

Figure 7: Processed top camera image. Grey lines indicate the extrapolated fiber best fit lines. White lines indicate the estimated fiber lateral displacement.

2.2.1.2. Cantilever Coupler Detection

In order to detect the position of the cantilever and pit in a side camera image, a template matching algorithm is used. A template image of a cantilever and pit is swept across the image, and the point on the image with the best match is determined to be the location of the pit. Using empirically determined offsets, the exact position of the cantilever is computed from the location of the pit. An image with vision processing indicators is shown in Figure 8. The detected fiber tip is highlighted in red, the detected location of the pit is highlighted in a green rectangle, and the estimated location of the cantilever is indicated with a green circle.

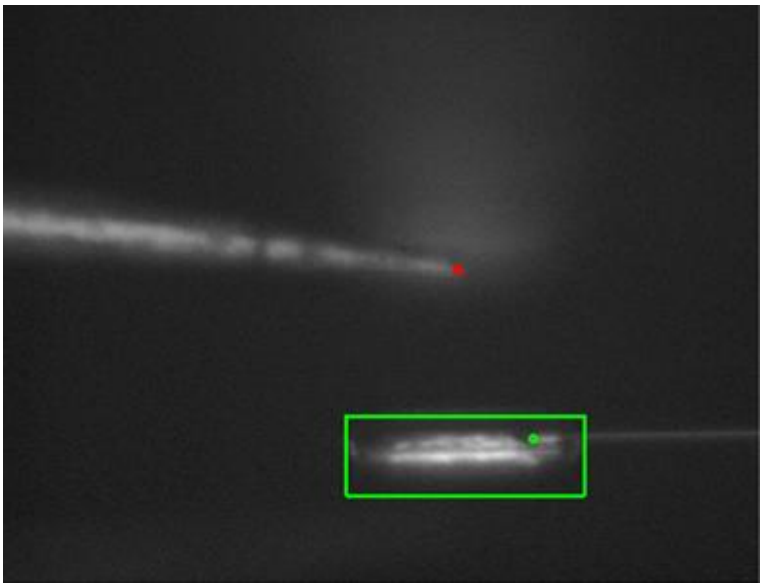

Figure 8: Processed side camera image. The detected fiber is marked with a red circle. The detected pit location is marked with a green rectangle, and the estimated cantilever location is marked with a green circle.

2.2.2. Coupling Algorithm

To begin the automated coupling algorithm for fiber-to-chip coupling, the human operator must first manually position the fibers in the input and output coupler pits. The manual micrometers on the fiber positioning stages can have a resolution of up to ~1 μm, which is

sufficient for this step. This manual positioning step is required since the piezoelectric controls have a limited range of 30 µm. This manual alignment serves as a rough calibration for alignment in the transverse axes (the *y*- and *z*-axes).

The next step in the algorithm is to measure the axial displacement between the fiber and the coupler through vision processing. In order to accurately measure this visually, the fiber is moved up 10 µm along the *x*-axis and another 10 µm laterally along the *y*-axis. This movement ensures that the fiber and cantilever coupler appear as separate features in the image, and that both of them are in focus. Next, the axial displacement between the fiber and the cantilever is repeatedly calculated using vision processing. Ten samples are collected and averaged to yield an accurate estimate. Once the displacement is computed, the fiber is positioned so that a distance of 5 µm remains between the fiber and its target along the *x*-axis. Controlling the axial displacement prevents unintended collisions between the fiber and the cantilever. After this, the fiber is moved back to its original manually calibrated position in the transverse plane (the *y*- and *z*-axes), resulting in a roughly aligned fiber with a 5 µm axial displacement.

Next, the fiber is more optimally aligned using coupled power measurements. A sweep of the fiber position in the transverse plane is performed, and coupled power is measured and recorded at each point. The results of such a sweep are shown in Figure 9, where the horizontal axes are the indices of fiber positions in the transverse plane, and the vertical axis is the coupled power with

an averaging filter applied. The point of maximum coupled power, indicated with a marker in Figure 9, corresponds to the fiber position that is most optimally aligned for coupling.

In order to improve algorithm speed, as well as to avoid unintended collisions with chip features while doing fiber-to-chip coupling, a sequence of 1-dimensional sweeps are performed to

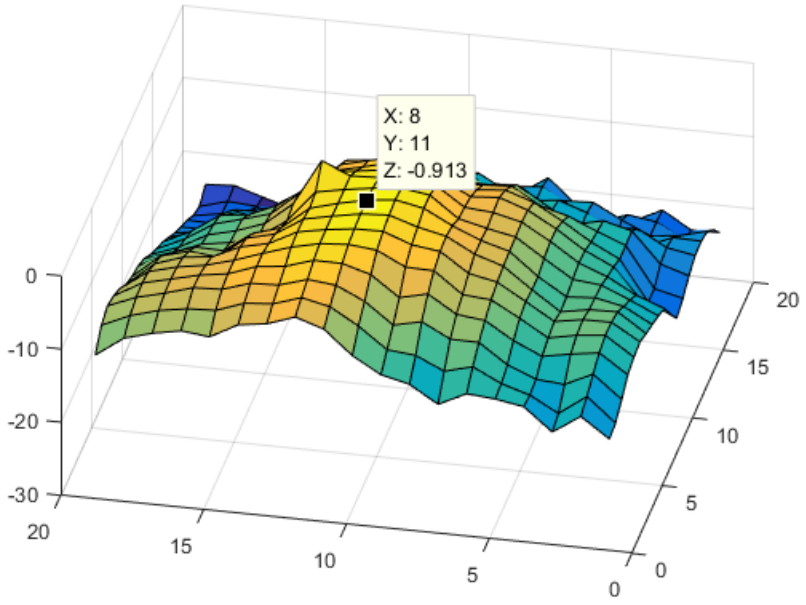

Figure 9: Coupled power for different fiber positions in the transverse plane. The coupled power has been filtered and normalized to the maximum measured power.

align the fiber for coupling. One such sweep is performed along the *y*-axis, followed by another along the *z*-axis. Since the shape of a constant power contour of the data in Figure 9 is approximately symetric, it can be modeled by decomposing it into two orthogonal 1-dimensional contours. Using this model, the point of maximum coupled power can be determined simply from the optimum positions along the *y*- and *z*-axes. Since vastly fewer data points are required for this method, the measurements taken along each axis are averaged across 10 samples. The averaged results of one such sweep along the *z*-axis are shown in Figure 10. Note that the assymetric nature of the measurements along the *z*-axis correspond to measurements with the fiber tip ranging from

below the chip surface to above it. The optimal fiber position, is computed by taking the simple maximum of the average coupled power. Other methods of computing optimum fiber alignment based on power meter scan data were investigated, including spacial filtering and a center-of-mass style calculation. The simple maximum was determined to be superior to these because it is reslient to measurement noise due to the averaging, and because it remains sensitive to locating maxima near the edge of the scan.

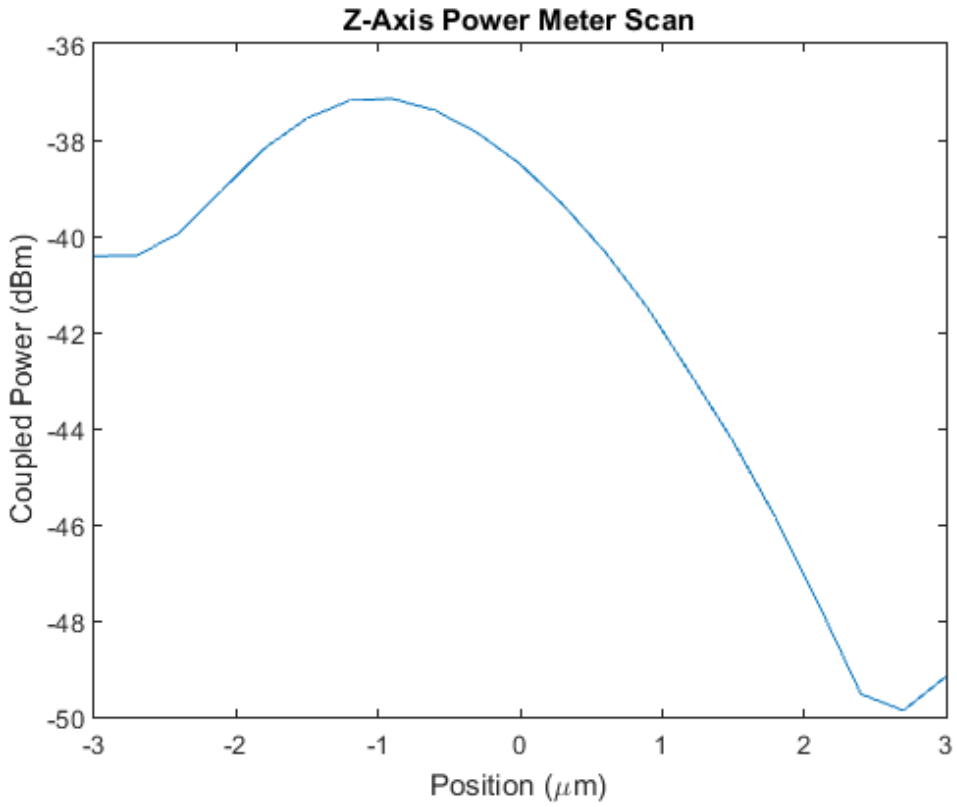

Figure 10: Average coupled power for different fiber positions along the *z*-axis. Here, the relative position along the *z*-axis where the fiber is optimally aligned is -0.9 µm.

Once the fiber has been roughly positioned axially using vision processing and roughly aligned through the manual calibration step, an initial power meter sweep is performed to determine a more optimially aligned fiber position based on measured coupled power. This initial sweep covers a 6 μm x 6 μm area with a resolution of 0.3 μm. Once the sweep has been performed, the fiber is moved to the optimal position that was calculated, and incremented along the *x*-axis toward the target fiber or cantilever. The size of this increment is dependent on the current estimated distance between the fiber and its target fiber or cantilever. This distance is estimated by combining the initial displacement estimated from the vision processing results with all subsequent movements of the *x*-axis piezoelectric stage. If the estimated axial distance is greater than 2 μm, the size of the fiber position increment is chosen to be 0.5 μm, and if it is under 2 μm, each increment will be 100 nm. After several such increments, once the fiber is within 2 μm of butt-coupling to the target fiber or cantilever, a second power meter sweep is performed in the transverse plane to bring the fiber into more optimal alignment for coupling. After the fiber position has been adjusted as a result of the second sweep, sucessive increments are performed as before along the *x*-axis, until coupling is achieved.

After each increment along the *x*-axis, the automated coupling system must determine whether butt-coupling has been achieved. It is not possible to determine this purely from the vision processing results, since visual displacement measurements are only reliable with ~1 μm resolution, and ~100 nm resolution would be required to accurately detect butt-coupling. Thus, the coupling detection algorithm uses a metric based on the stability of the measured coupled power over a period of two seconds. Due to fiber vibrations and noise, coupled power varies significantly more when coupling has not yet occurred than when the fibers are in contact. To detect these changes, the magnitude of the time derivative of the coupled power is computed and averaged

across the measurement period to yield a metric for coupling detection. In addition, since one port must be coupled before the other in a full 2-port coupling process, the coupling detection algorithm needs to be able to detect the transition from no fibers coupled to one fiber coupled, as well as the transition from one fiber coupled to two fibers coupled.

Initially, the coupling detection algorithm used hard-coded thresholds to detect the different states of coupling. This approach, however, has the disadvantage of being strongly dependent on the surrounding wind environment. In particular, the presence or absence of wind isolation around the fibers has a huge effect on the coupled power stability metric. Figure 11 shows the measured coupled power and its derivative for the system with and without wind isolation for the case where both fibers are uncoupled. The presence of wind isolation has the effect of

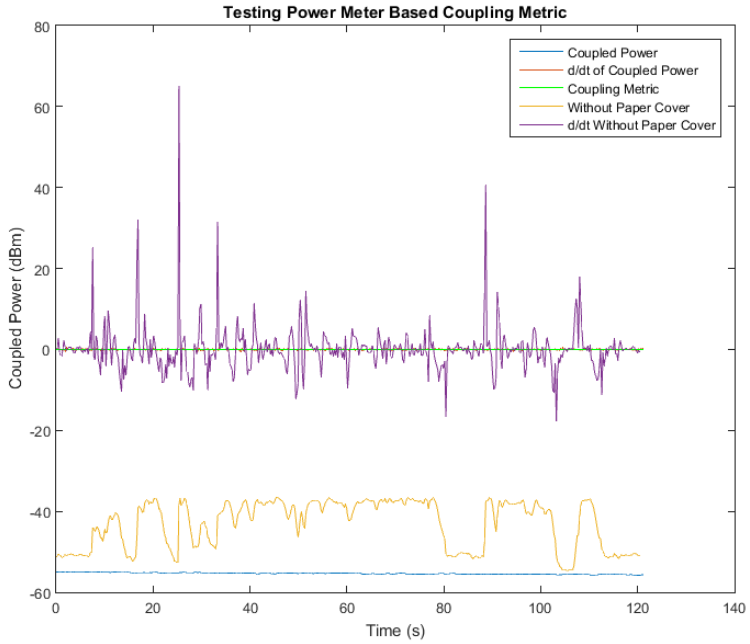

Figure 11: Measured coupled power and its time derivative with and without wind isolation. The yellow and purple traces show the coupled power and derivative without wind isolation, respectively. The blue and red traces show the coupled power and derivative with wind isolation, respectively.

stabilizing the coupled power and reducing the magnitude of its derivative, meaning hard coded thresholds cannot be robust to changes in the experimental environment.

While wind isolation is clearly a good way to reduce fiber vibrations and therefore reduce variation in the coupled power, this also leads to a reduction in the value of the derivative-based metric used for coupling detection. This makes it difficult to reliably detect coupling with hard-

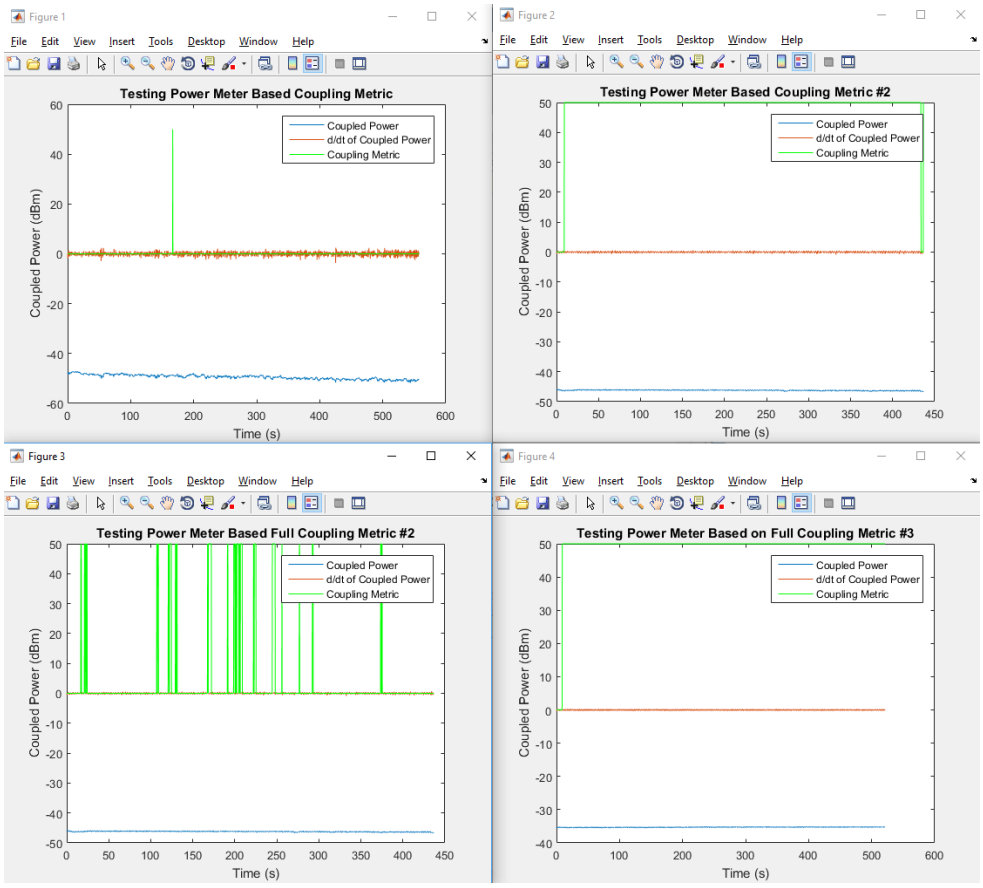

Figure 12: Using hard-coded thresholds to detect coupling. The blue traces are measured coupled power. The orange traces are the derivative of the coupled power. The green traces are high at times when the system detects coupled. Top-left: no fibers are coupled. Top-right: one fiber is coupled. Bottom-left: one fiber is coupled. Bottom-right: both fibers are coupled. The top two plots correspond to the transition from no fibers coupled to one fiber coupled, and the bottom two graphs correspond to the transition from one fiber coupled to two fibers coupled.

coded thresholds without getting false positives. Figure 12 shows the result of the coupling detection algorithm using hard-coded thresholds with wind isolation added to the system. Note that false positives are common, especially for the case where one fiber is coupled and the system is attempting the detect the transition to two fibers being coupled.

In order to address these problems, a relative coupling detection algorithm is used. An initial baseline metric is computed when the fiber is known to be un-coupled, and subsequent

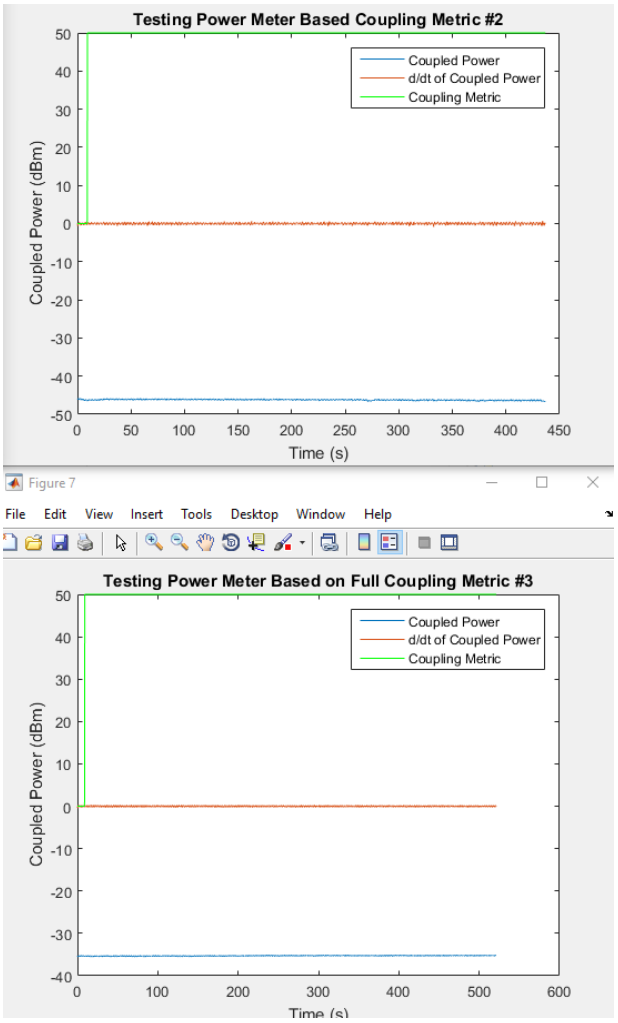

Figure 13: Using relative thresholds to detect coupling. The blue traces are measured coupled power. The orange traces are the derivative of the coupled power. The green traces are high at times when the system detects coupled. Top: one fiber is coupled. Bottom: both fibers are coupled.

measurements are compared to the baseline to determine if coupling has occurred. The metric must be below either 40% or 65% of the value of the baseline for the algorithm to detect a single fiber coupled and both fibers coupled, respectively. These threshold coefficients were determined empirically. Since the coupling metric must be significantly lower than the baseline metric, false positives are significantly reduced. Figure 13 shows the results of the coupling detection algorithm using this relative measurement. This relative measurement allows the system to be robust to changes in the environment, such as varying wind isolation. Continuing with the coupling algorithm, if the metric indicates that coupling has not occurred, the algorithm proceeds to increment the fiber position along the *x*-axis. If, however, the coupled power measurements indicate that coupling has occurred, the automated coupling system stops, and proceeds to couple the remaining fiber.

One of the main concerns with a fully automated butt-coupling system such as this is that a system error may result in damaging or breaking the chip or the fiber, particularly if butt-coupling is not properly detected and the system continues to press the fiber into the chip. The system has a safeguard to prevent this from occurring. The original estimate of the axial displacement that was computed from the vision processing results is used to continually estimate the remaining axial displacement between the fiber and cantilever. If the system does not determine that butt-coupling has occurred within ±2 μm of this vision processing-based estimate, it stops the algorithm and restarts the coupling process.

For full device scale coupling, the input fiber is first coupled to the input cantilever. Then, once the cameras have been moved to view them, the output fiber is coupled to the output cantilever. The automated coupling process takes approximately 9 minutes to couple each fiber to its respective cantilever. Additionally, before each movement of the piezoelectric actuators, the

human operator is given the option to proceed with or stop the automatic coupling algorithm. They can choose to override this feature by selecting a checkbox that enables fully automatic operation.

2.3. Coupling Algorithm Results

The system is capable of automatically performing both fiber-to-fiber and fiber-to-chip coupling. The following figures will illustrate the process of coupling the input fiber to the on-chip

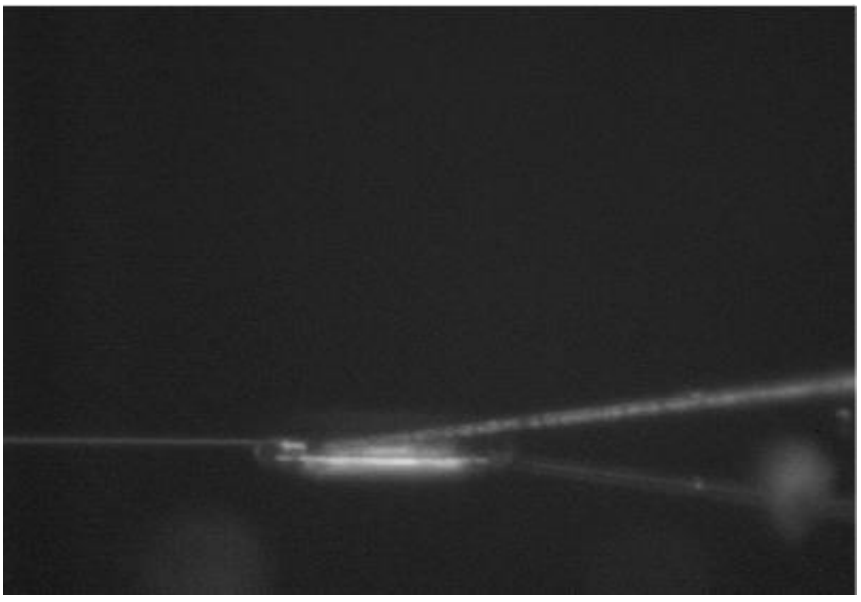

Figure 14: The side-mounted camera image after the operator manually positions the fiber in the pit.

cantilever coupler. Figure 14 shows the position of the fiber after the human operator roughly aligned the fiber with the cantilever.

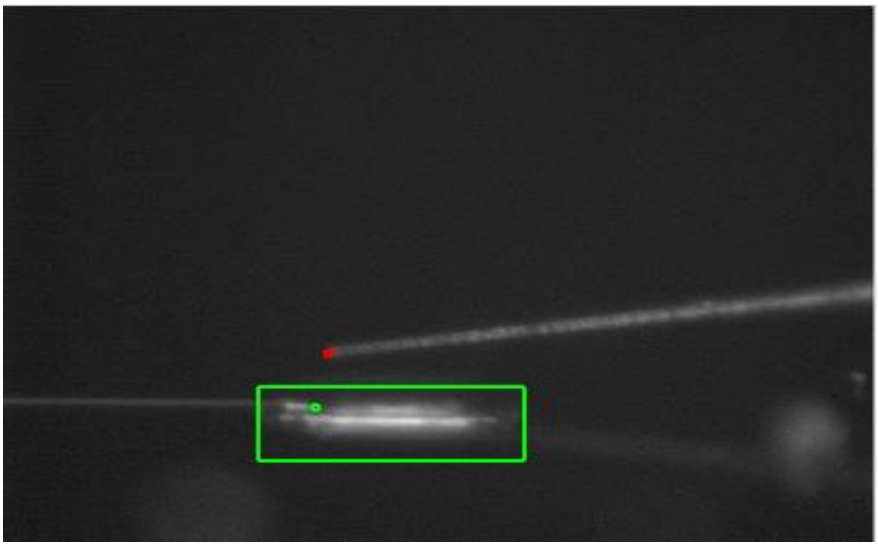

Figure 15: The side-mounted camera image after the fiber and cantilever are visually detected.

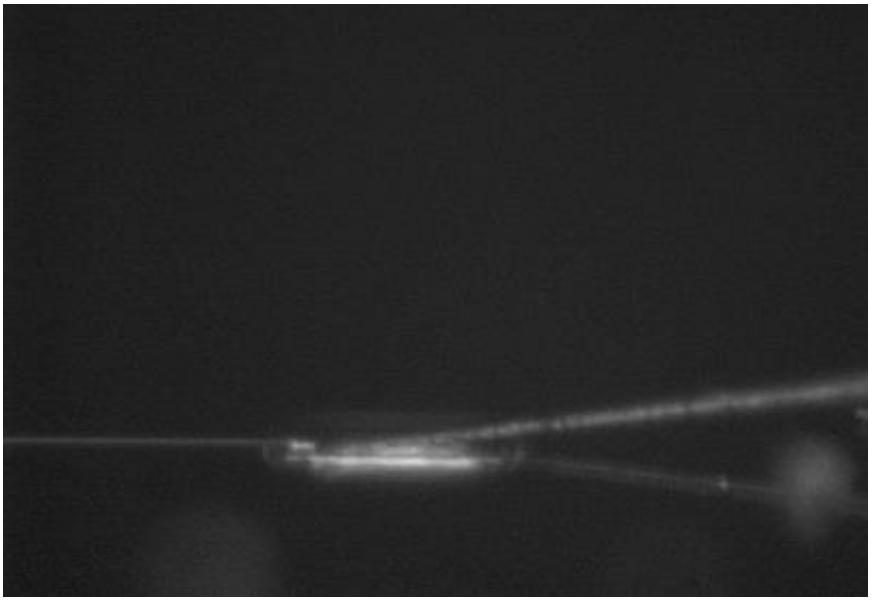

Figure 16: The side-mounted camera image after the fiber has been aligned using the first power meter scan.

Figure 15 shows the output of the vision processing algorithm used to determine the axial displacement of the fiber and cantilever. The detected position of the fiber tip is highlighted in red, while the detected position of the cantilever is shown in green. After visual detection, the system performs a power meter scan and moves the fiber to the optimal position for alignment. Figure 16 shows the position of the fiber after this alignment.

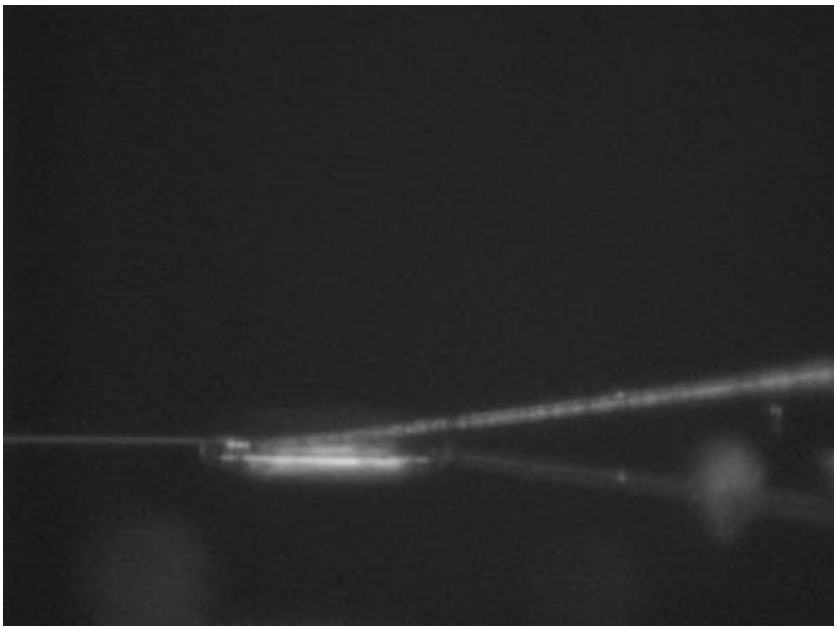

Figure 17: The side-mounted camera image once the input fiber has been coupled to the cantilever.

After additional power meter sweeps, the fiber is brought into contact with the cantilever. This result is shown in Figure 17. After this, the system repeats the process on the output fiber, the result of which can be seen in Figure 18.

The system is also capable of providing various degrees of assistance to a manual operator, including vision processing feedback for alignment, electronic control of stages, and manual execution of the coupling algorithm.

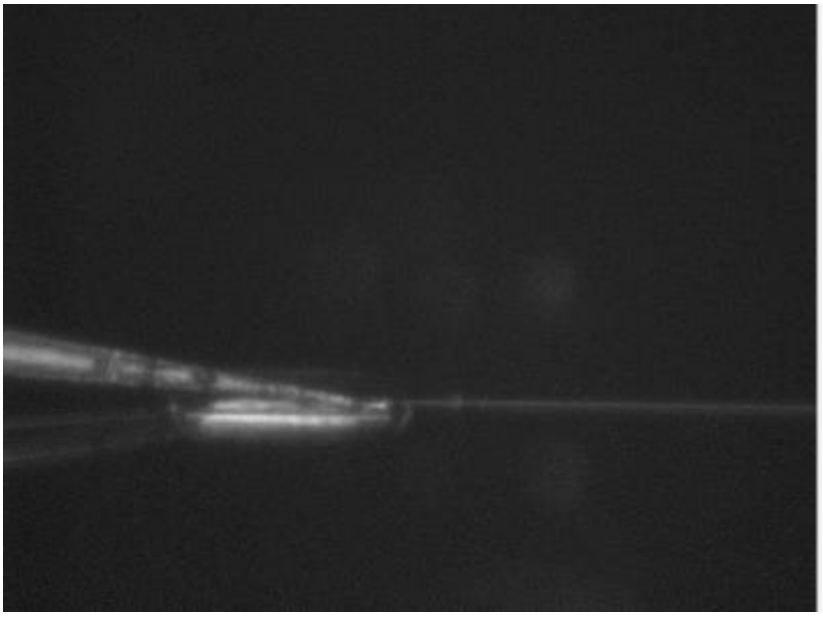

Figure 18: The side-mounted camera image after the output fiber has been coupled to its corresponding cantilever.

2.4. Variability of Coupled Power

The automatic coupling system does not currently achieve the desired power coupling efficiency or power coupling consistency. In a test of 11 fiber-to-fiber coupling attempts, the mean measured final coupled power was -21.4 dBm, compared to the theoretical maximum of -17.8 dBm. Moreover, the standard deviation of the coupled power of these 11 attempts was 2.9 dB. A number of sources of error prevent perfectly accurate coupling on every attempt. The most significant factor that limits consistency in coupled power is fiber vibration caused by drafts of air interacting with the fiber. Vibrations with a range as large as 1 μm can be visually seen on the camera feed. Since this is on the order of the size of the fiber and cantilever tips, this has a large effect on the final coupled power.

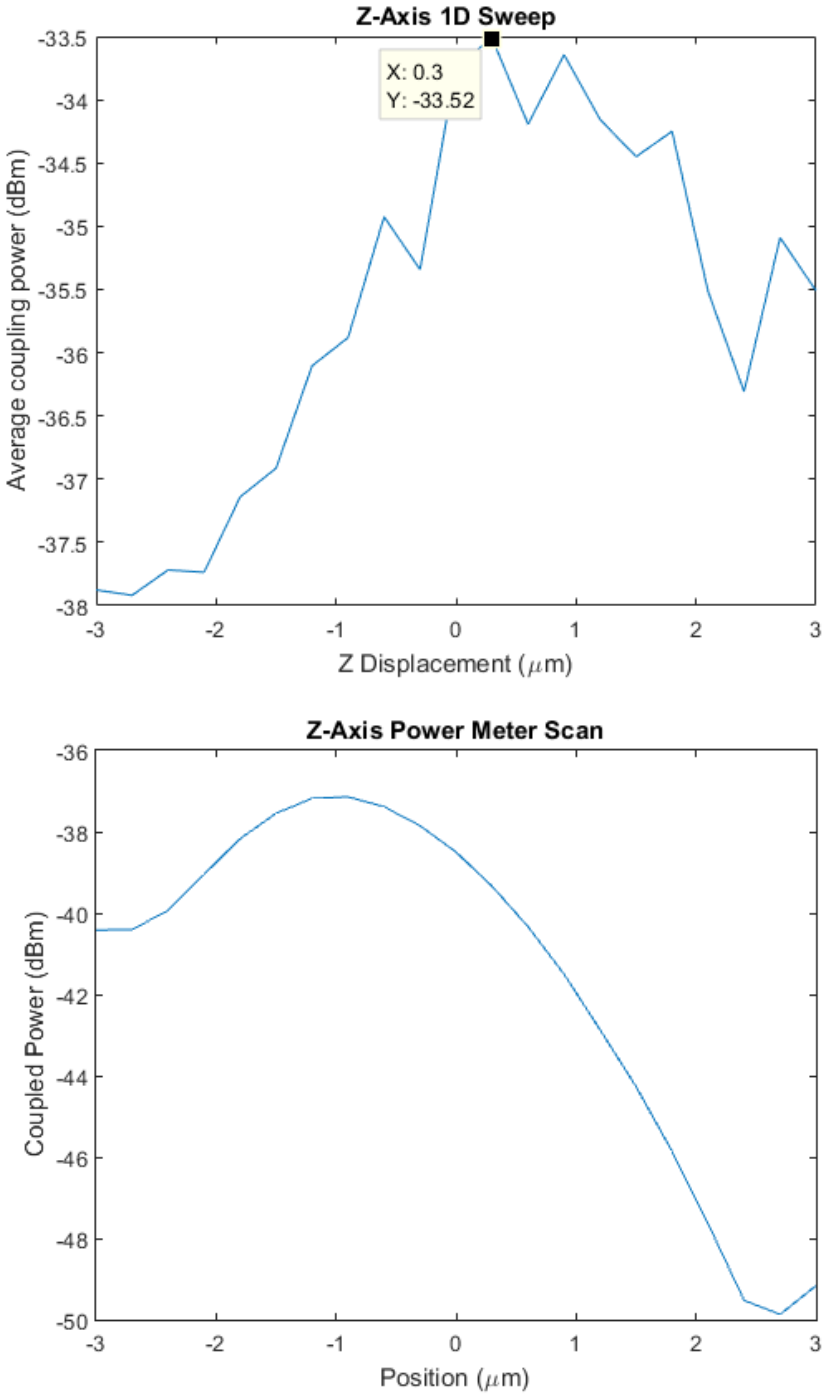

Figure 19: *Z*-axis power meter scans illustrating the effect of external air drafts on fiber vibrations. The noisy data shown in the top plot was collected with the normal system. The smooth data shown in the bottom plot was collected with an enclosure isolating the fibers and chip from the surrounding air.

These vibrations were greatly reduced by isolating the fibers from any outside air drafts. Figure 19 shows two plots of power meter measurements along the *z*-axis. Even with sample averaging, the data in the top plot in Figure 19 is noisy, indicating large fiber vibrations that can cause poor coupler efficiency and consistency. The bottom plot in Figure 19, by contrast, contains only minimal noise since the fibers and chip were better isolated from external air drafts.

3. Results

3.1. Measurement Calibration

In order to accurately measure the coupling efficiency that the automated coupling system can achieve, the experimental setup must first be calibrated to compensate for losses in the optical interconnects surrounding the cantilever couplers and the device under test. This calibration is accomplished by measuring the maximum coupled power in a fiber-to-fiber measurement. This baseline power can then be subtracted from subsequent measurements in order to isolate the insertion loss of the cantilever couplers and the device under test.

For this system, the maximum measured coupled power for fiber-to-fiber coupling was -17.8 dBm. The tunable laser was set to output light at -10 dBm, so the losses in the fibers and connectors external to the device under test were determined to be 7.8 dBm. Note that this figure includes coupling losses between the two tapered fibers. Although the fiber tapers are a component of the overall cantilever coupler system, these losses can be neglected for sufficiently long tapers. The maximum coupled power was determined by performing fiber-to-fiber coupling multiple times, and at various degrees of alignment in the transverse plane. For this measurement, the tapered fibers were rotated so that they were axially aligned with each other, as shown in Figure 20.

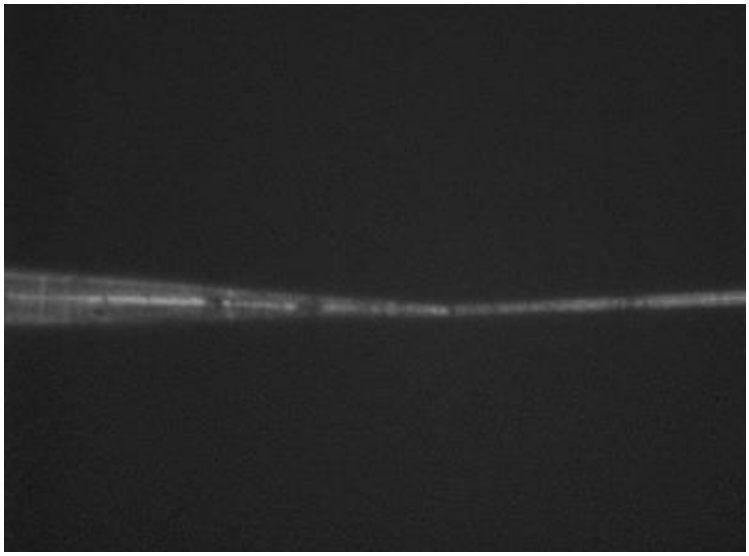

Figure 20: Side-mounted camera view of the result of the fiber-to-fiber coupling operation used for system calibration.

The measured coupled power for the fiber-to-fiber coupled system across multiple relative alignments in the transverse plane is shown in Figure 21. As can be seen from the histogram, the maximum coupled power was -17.8 dBm, and the majority of data points are near this value.

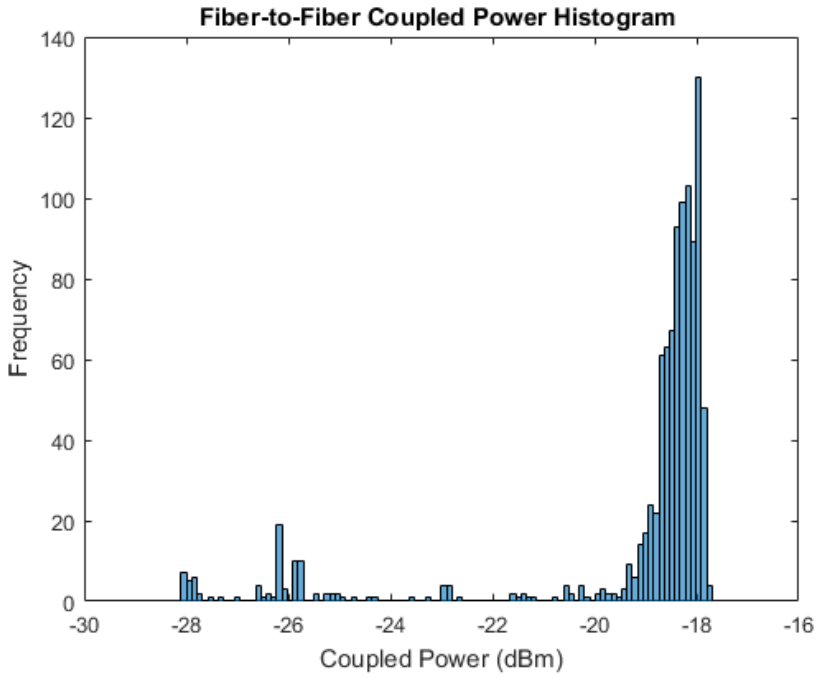

Figure 21: Histogram of measured coupled power for fiber-to-fiber calibration.

3.2. Coupled Power Measurements

The automated coupling system's capability for fiber-to-chip coupling was tested on a straight waveguide device with cantilever couplers at the input and output. These measurements were performed with wind isolation added to the system. The length of the waveguide was 500 μm, which for an estimated propagation loss of 9.5 dB/cm [3] yields a device insertion loss of 0.45 dB. This value was subtracted from the final coupled powers along with the -17.8 dBm baseline to isolate the insertion loss of both cantilever couplers. For a series of four complete automatic

coupling processes, the average insertion loss of both couplers combined was 8.9 dB, and the standard deviation was 3.8 dB. The minimum measured insertion loss was 3.6 dB, which corresponds to an insertion loss of 1.8 dB per cantilever coupler. This coupling efficiency means that the automated coupling system is capable of approaching the ideal performance of the cantilever couplers, as reported in [3]. However, the variation in the final coupling power is quite large, and requires additional work to improve.

4. Future Work

4.1. Reducing Coupling Variation

The automated coupling system currently has a large variation in the final coupled power, largely due to random fiber vibrations. Due to the presence of these fiber vibrations, even if they are dramatically reduced, it is useful to consider the position of the fiber in space as a probability distribution centered on the controllable position of the piezoelectric stages. Once butt coupling has been achieved, however, the fiber is held in place at a single point. Therefore, the automated coupling algorithm has been extended to include multiple attempts at coupling. Since any one attempt to couple can statistically result in poor coupling power, multiple attempts must be made, until an acceptable level of coupled power is reached. Additional work needs to be done to implement a way to automate the decision to stop coupling at an acceptably high coupled power.

4.2. Automatic Testing System

Once the system can perform fully automated fiber-to-chip coupling with suitably low variation in final coupled power, a fully automated testing system for devices with cantilever couplers can be created. In order to achieve this, the reliability of device scale automatic coupling

needs to be improved, and then chip scale automatic coupling need to be implemented. Then, devices across an entire chip will be able to be tested automatically. Optical transmission measurements can be performed over wavelength and power sweeps using the tunable laser, and electro-optical measurements can also be incorporated into the system.

In order to improve the automatic device-scale coupling algorithm, the coupling detection algorithm needs to be enhanced to be resilient to various amounts of environmental noise. Additionally, the algorithm to attempt butt-coupling multiple times in a row needs to be more fully automated. Later on, the chip itself can be mounted on a 2-axis motorized stage to translate between devices on the same chip, enabling automatic testing of all devices on the same chip.

The goal of this work is to provide a platform upon which a fully automated testing system capable of testing multiple devices on the same chip can be created. This system can save time for researchers, reduce the training and expertise required to perform tests, and improve the accuracy and consistency of measurements. Additionally, an automated testing platform for high performance cantilever couplers would enable their broader use in research and manufacturing testing applications.

5. References

[1] Mekis, Attila, Steffen Gloeckner, Gianlorenzo Masini, Adithyaram Narasimha, Thierry Pinguet, Subal Sahni, and Peter De Dobbelaere. "A Grating-Coupler-Enabled CMOS Photonics Platform." IEEE J. Select. Topics Quantum Electron. IEEE Journal of Selected Topics in Quantum Electronics 17.3 (2011): 597-608. Web.

[2] Peng Sun and Ronald M. Reano, "Cantilever couplers for intra-chip coupling to silicon photonic integrated circuits," Opt. Express 17, 4565-4574 (2009)

[3] Michael Wood, Peng Sun, and Ronald M. Reano, "Compact cantilever couplers for low-loss fiber coupling to silicon photonic integrated circuits," Opt. Express 20, 164-172 (2012)

[4] Yunhong Ding, Haiyan Ou, and Christophe Peucheret, "Ultrahigh-efficiency apodized grating coupler using fully etched photonic crystals," Opt. Lett. 38, 2732-2734 (2013)

[5] Chao Li, Huijuan Zhang, Mingbin Yu, and G. Q. Lo, "CMOS-compatible high efficiency double-etched apodized waveguide grating coupler," Opt. Express 21, 7868-7874 (2013)

[6] Tomoya Yoshida, Syougo Tajima, Ryohei Takei, Masahiko Mori, Noboru Miura, and Youichi Sakakibara, "Vertical silicon waveguide coupler bent by ion implantation," Opt. Express 23, 29449-29456 (2015)

[7] Peng Sun and Ronald M. Reano, "Vertical chip-to-chip coupling between silicon photonic integrated circuits using cantilever couplers," Opt. Express 19, 4722-4727 (2011)

[8] Jeroen De Coster, Peter De Heyn, Marianna Pantouvaki, Brad Snyder, Hongtao Chen, Erik Jan Marinissen, Philippe Absil, Joris Van Campenhout, and Bryan Bolt, “Test-station for flexible semi-automatic wafer-level silicon photonics testing,” in *2016 21st IEEE European Test Symp. (ETS)*, (2016)

Appendix A: User Interface and Operator Instructions

User Interface

A screenshot of the GUI for the automatic coupling system is reproduced in Figure 22, with additional markings to identify each of the 12 sections of the interface. The function of each of these sections will now be described in detail:

1. The mode selection section has a drop down menu that allows the user to select from a number of modes of operation. The "Vision Processing (F2C)" and "Vision Processing (F2F)" modes cause the system to output the results of the vision processing algorithm for fiber-to-chip detection or fiber-to-fiber detection, respectively. The "Automatic Fine Coupling (F2C)" and "Automatic Fine Coupling (F2F)" modes cause the system

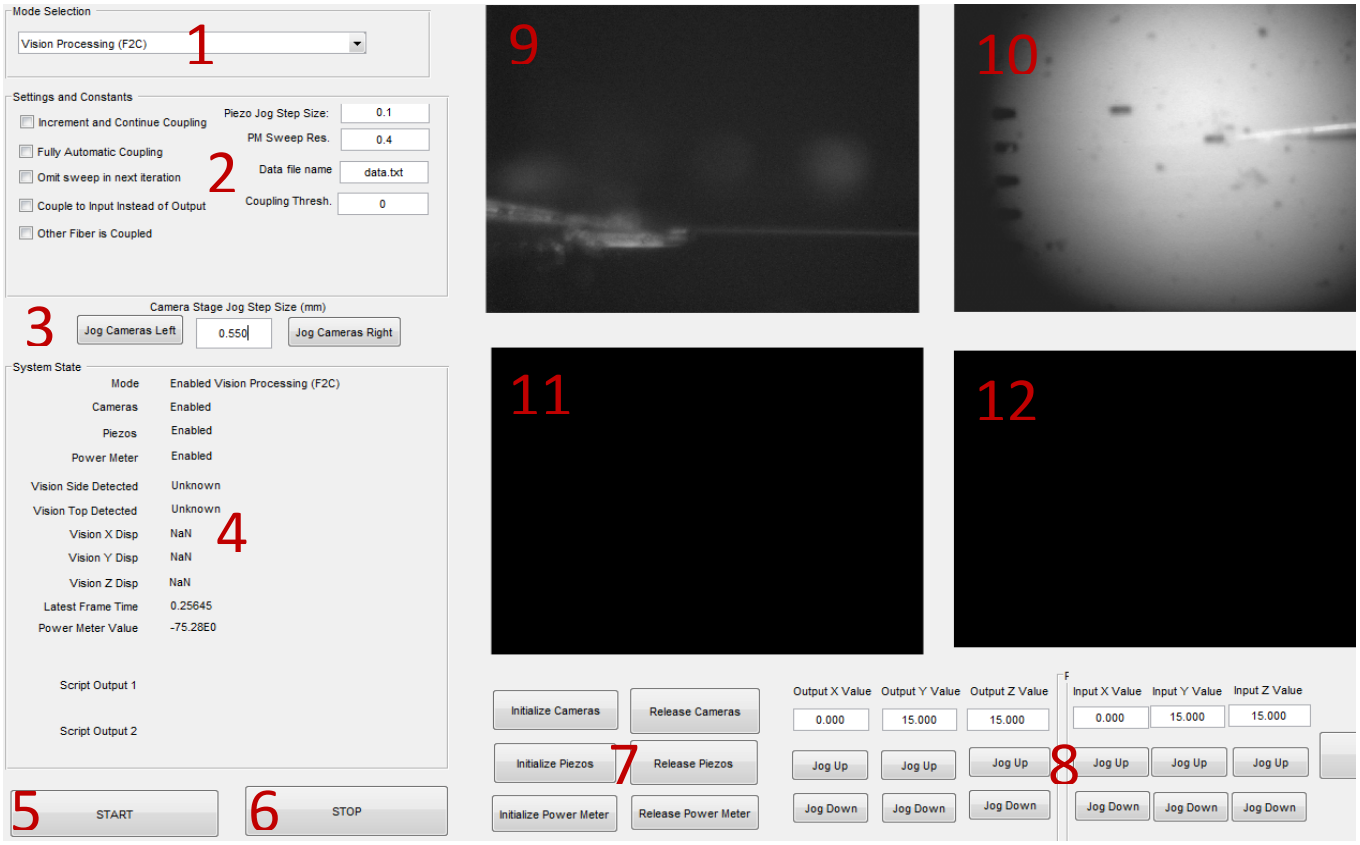

Figure 22: General User Interface (GUI) for the automated coupling system, with various sections marked.

to perform the automated coupling algorithm for one-port fiber-to-chip coupling or fiber-to-fiber coupling, respectively. The “Automatic Full Device Coupling” mode caused the system to perform the full two-port fiber-to-chip coupling algorithm. The “Power Meter Scan” mode causes the system to perform a power meter scan independent of a full coupling algorithm. Finally, the “Check if Coupled” mode causes the system to perform the coupling detection algorithm independent of a full coupling algorithm.

2. The settings and constants section provides multiple options and settings that the user can set to change the behavior of the automated coupling system. The following is a list of these settings:
    - The “Increment and Continue Coupling” checkbox is used as a way for the user to approve the next part of the coupling process. At various points in the automated coupling process, the system will pause and wait for the user to check this box before continuing with the algorithm.
    - The “Fully Automatic Coupling” checkbox is used to enable fully automatic coupling without need for user approval throughout the process. When this is checked, the system will consider each action automatically approved, and will not pause execution during the coupling process. While this checkbox is checked, there is no need for the user to use the “Increment and Check if Coupled” checkbox.
    - The “Omit sweep in next iteration” checkbox is used to bypass any power meter sweeps in the automated coupling process. While this checkbox is checked, the automated coupling system will not perform any power meter sweeps that are

part of a full coupling algorithm. The user can use this feature to avoid time-intensive power meter scans if they are confident that the fiber is already aligned.

- The “Couple to Input Instead of Output” checkbox serves as a switch between coupling to the input cantilever coupler port and coupling to the output cantilever coupler port. This setting needs to be set manually when performing one-port coupling, but will be set automatically when running the full two-port coupling algorithm.
- The “Other Fiber is Coupled” checkbox serves as a switch to indicate to the system whether or not the other fiber in the system has been coupled to its cantilever coupler. This information is used in the coupling detection algorithm. This setting needs to be set manually when performing one-port coupling, but will be set automatically when running the full two-port coupling algorithm.
- The “Piezo Jog Step Size” text input field allows the user to specify the distance that each piezoelectric actuator will move when the user manually jogs it. This setting will not affect the automatic coupling algorithms. The units for this field are in μm and the range of acceptable values for this field is (0.0, 30.0], although the piezoelectric actuators only have an accurate resolution of 0.1 μm.
- The “PM Sweep Res.” Text input field allows the user to specify the resolution, in μm of the next power meter sweep performed by the system. This only needs to be set manually prior to running the “Power Meter Scan” mode; the automatic coupling algorithms will update this field with the appropriate resolution as necessary.

- The “Data file name” text input field allows the user to set the name of the text file that will contain the data collected by subsequent power meter scans for diagnostic purposes.
- The “Coupling Thresh” text input field contains the baseline metric used in the coupling detection algorithm. This only needs to be set manually if the user is running the “Check if Coupled” mode; the automatic coupling algorithms will update this field automatically with the appropriate baseline value.

3. This section is used to provide manual controls for moving the camera stage. The two buttons allow the user to jog the camera stage left and right. The “Camera Stage Jog Step Size (mm)” text input field allows the user to set the distance that the camera stage will move once jogged. This field must be set accurately before running the full two-port coupling algorithm, since the algorithm relies on the manually determined jog distance to move from the input coupler to the output coupler.
4. The system state panel provides different types of feedback to the user. The following is a list of the displayed information:
    - “Mode” displays the name of the current mode of operation, or “Disabled.”
    - “Cameras” displays whether or not the cameras have been initialized.
    - “Piezos” displays whether or not the piezoelectric actuators have been initialized.
    - “Power Meter” displays whether or not the power meter has been initialized.
    - “Vision Side Detected” displays the features that vision processing has detected in the side-mounted camera image.

- "Vision Top Detected" displays the features that vision processing has detected in the top-mounted camera image.
- "Vision X Disp" displays the displacement along the *x*-axis computed from vision processing. This estimate will update continually during the automated coupling process.
- "Vision Y Disp" displays the displacement along the *y*-axis computed from vision processing.
- "Vision Z Disp" displays the displacement along the *z*-axis computed from vision processing.
- "Power Meter Value" displays the current coupled power, in dBm.
- "Script Output 1" and "Script Output 2" are generic fields that the automated coupling system can use to display information specific to a mode of operation.

5. The "START" button is used to begin the mode of operation that is currently selected.
6. The "STOP" button is used to stop the current mode of operation, and can be pressed to stop the coupling process at any time. After this button is pressed, the system will enter "Disabled" mode, where only the live camera feed and the power meter data are displayed.
7. This section is used to initialize and release the three major subsystems of automated coupling system: the cameras, the piezoelectric actuators, and the power meter. These subsystems need to be initialized before running an automated coupling algorithm, and released before closing the user interface. Note: When the piezoelectric actuators are initialized, they will move immediately. They will also move when they are released. For this reason, the operator must be sure that the fibers are a safe distance (>30 μm)

away from each other and from the chip surface before either initializing or releasing them.

8. This section allows the user to manually control the piezoelectric actuators by jogging each of the axes on either the input or output fiber up or down. The distance that each fiber will move when jogged is determined by the "Piezo Jog Step Size" setting. Additionally, text fields in this section display the current position, in μm, of each of the piezoelectric axes.
9. This section is where the camera feed from the side-mounted camera is displayed, along with any results from the vision processing output.
10. This section is where the camera feed from the top-mounted camera is displayed, along with any results from the vision processing output.
11. This section is typically empty, but can be used to display the thresholded version of the side-mounted camera feed for diagnostic purposes.
12. This section is typically empty, but can be used to display the thresholded version of the top-mounted camera feed for diagnostic purposes.

Operator Instructions

The following is the procedure an operator must follow to perform a full two-port coupling process:

1. Initialize the cameras.
2. Run the "Vision Processing (F2C)" mode to view the camera feeds and vision processing feedback.

3. Initialize the piezoelectric actuators. Make sure that both fibers are a safe distance away from everything before initializing them.
4. Initialize the power meter.
5. Manually align the first fiber with its cantilever coupler, as shown in Figure 14.
6. Set the “Camera Stage Jog Step Size (mm)” field to the appropriate distance between the input and output cantilever couplers. This value can be determined empirically or from the device schematic.
7. Jog the camera stage to bring the second fiber and cantilever into view.
8. Manually align the second fiber with its cantilever coupler, as shown in Figure 14.
9. If not already there, jog the camera stages to bring the input fiber input view.
10. If desired, change the “Data file name” field to an appropriate name.
11. Run the “Automatic Full Device Coupling” mode.
12. If desired, check the “Fully Automated Coupling” checkbox to enable the fully automated coupling process. Otherwise, check the “Increment and Continue Coupling” checkbox whenever the system pauses and requests user input.
13. Once the system is finished, the operator can choose to manually move the fibers and optimize the final coupled power.
14. Perform device tests.
15. Move the fibers back along the *x*-axis to uncouple them.
16. Manually move the fibers a safe distance from the chip surface, and release the piezoelectric actuators.
17. Release the power meter.

18. Release the cameras.

19. Close the GUI.

The procedure an operator must follow for one-port fiber-to-chip coupling is the same as that for two-port coupling, with a few exceptions. The “Automatic Fine Coupling (F2C)” mode must be run instead of the two-port coupling mode, and the camera stage can be moved to either the input or the output fiber before running the algorithm. The “Coupled to Input Instead of Output” checkbox needs to be set to correspond to the correct coupler, and the “Other Fiber is Coupled” checkbox needs to be set to correspond to whether or not the other fiber has previously been coupled to its cantilever.

The procedure an operator must follow for fiber-to-fiber coupling is also the same as that for two-port coupling, with a few exceptions. The camera stage should be moved so that both fibers are in view, and the fibers should be manually aligned with each other instead of roughly aligned in the pit. Once this is done, the “Automatic Fine Coupling (F2F)” mode needs to be run.

Appendix B: Computer Code

One-Dimensional Power Meter Scan Algorithm:

```
averaging = 10;
increment = str2double(get(handles.user_text_pm_sweep_resolution, 'String'));
range = increment * 10; % um, one sided

y_data = zeros(2*range/increment + 1, averaging);

if strcmp(get(handles.text_display_power_meter, 'String'), 'Enabled') &&
strcmp(get(handles.text_display_piezos, 'String'), 'Enabled')

    if get(handles.checkbox_couple_to_input, 'Value')
        [ret, base_y] = input_y_ctrl.GetPosOutput(0, 0);
    else
        [ret, base_y] = output_y_ctrl.GetPosOutput(0, 0);
    end

    if base_y - range < 0 || base_y + range > 30
        disp('Error: Range outside of piezo motion range. Exiting pm
scan...');
        return;
    end
    y_index = 21;
    averaging_iteration = 1;
    direction = -1;

    % Move to the edge of the testing range
    for i=1:10
        if get(handles.checkbox_couple_to_input, 'Value')
            input_y_ctrl.SetPosOutput(0, base_y + increment * i);
        else
            output_y_ctrl.SetPosOutput(0, base_y + increment * i);
        end

        draw_camera_and_pm;
        pause(0.25)
    end

    % Loop to collect data
    while ~strcmp(get(handles.text_display_mode, 'String'), 'Disabled')
        % First store the data at the current location
        fprintf(power_meter, 'R_A?');
        power = fscanf(power_meter);
        y_data(y_index, averaging_iteration) = str2double(power);

        % Then move to the next location
        if (y_index ==  1 && direction == -1) || (y_index == 21 && direction
== 1)
            if averaging_iteration >= averaging
                break
            end
```

```
            direction = -direction;
            averaging_iteration = averaging_iteration + 1;
        else
            y_index = y_index + direction;
            if get(handles.checkbox_couple_to_input, 'Value')
                input_y_ctrl.SetPosOutput(0, base_y + (y_index - 11) *
increment);
            else
                output_y_ctrl.SetPosOutput(0, base_y + (y_index - 11) *
increment);
            end
        end

        draw_camera_and_pm;
        pause(0.1);
    end

    % Save Output data
    fname = strcat('y_', get(handles.user_text_data_file_name, 'String'));
    f = fopen(fname, 'w');
    for i=1:size(y_data, 1)
        fprintf(f, '%f', y_data(i, 1));
        for j=2:size(y_data, 2)
            fprintf(f, ',%f', y_data(i, j));
        end
        fprintf(f, '\n');
    end
    fclose(f);

    % Analyze that data
    y_data_averaged = [];
    for i =1:size(y_data, 1)
        y_data_averaged(i) = mean(y_data(i, :));
    end
    [best, index] = max(y_data_averaged);
    optimal_val = (index - 11) * increment;
    disp('Optimal y increment:');
    disp(optimal_val);

    % Move to optimal y position
    if get(handles.checkbox_couple_to_input, 'Value')
        set(handles.user_text_input_piezo_y_value, 'String', num2str(base_y +
optimal_val));
    else
        set(handles.user_text_piezo_y_value, 'String', num2str(base_y +
optimal_val));
    end
    mode_move_piezos;
    pause(0.25);

end
```

Coupling Detection Algorithm:

```
if strcmp(get(handles.text_display_power_meter, 'String'), 'Enabled')

    pause(0.3);

    %clear the buffer
    for i = 1:10
        fprintf(power_meter, 'R_A?');
        power_data = str2num(fscanf(power_meter));
        pause(0.01);
    end

    NUM_SAMPLES = 200;
    data = zeros(NUM_SAMPLES, 2);
    for i=1:NUM_SAMPLES
        fprintf(power_meter, 'R_A?');
        power_data = str2num(fscanf(power_meter));
        data(i, 1) = toc;
        data(i, 2) = power_data;
        if exist('pm_data', 'var')
            pm_data(length(pm_data) + 1, :) = [toc, power_data];
        end
        pause(0.01)
    end

    % Analyze the power in the derivative of the collected data
    data_prime = data;
    data_prime(1, 2) = 0.0;
    for i = 2:size(data_prime, 1)
        data_prime(i, 2) = (data(i, 2) - data(i - 1, 2)) / (data(i, 1) -
data(i - 1, 1));
    end
    NUM_AVG = 40;
    metric = zeros(size(data_prime, 1), 1);
    metric(1:(NUM_AVG - 1)) = inf;
    for i = NUM_AVG:size(data_prime, 1)
        a = 0;
        for j = 0:(NUM_AVG-1)
            a = a + abs(data_prime(i - j, 2) * (NUM_AVG - j))^2;
        end
        metric(i) = a / (NUM_AVG^2-NUM_AVG) * 2*0.5;
    end
    baseline = mean(abs(data_prime(:, 2)).^1);
    metric = baseline

    % Choosing the threshold for coupling detection
    if get(handles.checkbox_other_fiber_coupled, 'Value')
        coupling_threshold_coefficient = 0.65; % Other fiber is already
coupled (10 for uncovered, 0.02 for covered (untested))
    else
        coupling_threshold_coefficient = 0.4; % Other fiber is not already
coupled (20 for uncovered, 0.05 for covered)
    end
    % Get the baseline coupling threshold from the previously computer
    % baseline, and multiply by a factor
    coupling_threshold =
str2num(get(handles.user_text_coupled_stddev_threshold, 'String'));
```

```
    is_coupled = metric < coupling_threshold *
coupling_threshold_coefficient;
    disp('metric7')


    if is_coupled
        % We are coupled!
        set(handles.text_display_script_output_1, 'String',
strcat('Coupled!'));
    else
        % We are not coupled...
        set(handles.text_display_script_output_1, 'String', strcat('Not
Coupled...'));
    end
    set(handles.text_display_script_output_2, 'String', strcat('Metric: ',
num2str(metric)));
    average_power = mean(data(:, 2))

end
```

Side Camera Vision Processing Algorithm:

```
function [ detected, fiber_positions, thresh, pit_position ] =
process_side_image_f2c( img, is_input )
%{
% @param img the preprocessed side camera image
% @param is_input a flag for whether the input or output port is being
coupled
%
% @return detected What was detected in the image. Can be:
%   - DETECTED_UNKNOWN
%   - DETECTED_TWO_FIBERS
%   - DETECTED_COUPLED_FIBERS
%
% @return fiber_positions Array of [x, y] pairs describing the positions
%                         of the fibers (input, then output fiber). This is
only returned if
%                         two fibers are detected. Otherwise, an empty list
is returned
%
% @return thresh The thresholded image for display purposes
%}

fiber_positions = [];
pit_position = [];
detected = Vision.DETECTED_UNKNOWN;

thresh = cv.adaptiveThreshold(  img, 255, ...
                                'AdaptiveMethod', 'Mean', ...
                                'ThresholdType', 'Binary', ...
                                'BlockSize',
VisionSide.ADAPTIVE_THRESH_KERNEL_SIZE, ...
```

```
                                    'C',
VisionSide.ADAPTIVE_THRESH_MEAN_MODIFIER);

thresh = cv.morphologyEx(thresh, 'Open', 'Element',
VisionSide.OPENING_KERNEL_SIZE);
thresh = cv.morphologyEx(thresh, 'Close', 'Element',
VisionSide.CLOSING_KERNEL_SIZE);

contours = cv.findContours(thresh, 'Mode', 'List', 'Method', 'Simple');

if length(contours) > 0

    blobs = cell(length(contours), 2);
    for i=1:length(contours)
        blobs(i, :) = {contours{i}, cv.contourArea(contours{i})};
    end
    blobs = flipud(sortrows(blobs, 2));
    contours = blobs(:, 1)';

    input_fiber_index = -1;
    output_fiber_index = -1;
    coupled_fiber_index = -1;
    for i=1:length(contours)
        hull = cv.convexHull(contours{i});
        area = cv.contourArea(hull);
        rect = cv.boundingRect(hull);
        aspect_ratio = rect(3) / rect(4);

        if      area > VisionSide.FIBER_SIZE_THRESHOLD * VisionSide.WIDTH *
VisionSide.HEIGHT & ...
                aspect_ratio > VisionSide.FIBER_ASPECT_RATIO_THRESHOLD & ...
                rect(1) > VisionSide.FIBER_NEAR_EDGE_THRESHOLD *
VisionSide.WIDTH & ...
                rect(1) + rect(3) > (1 -
VisionSide.FIBER_NEAR_EDGE_THRESHOLD) * VisionSide.WIDTH

            input_fiber_index = i;
            if detected == Vision.DETECTED_OUTPUT_FIBER
                detected = Vision.DETECTED_TWO_FIBERS;
                break
            else
                detected = Vision.DETECTED_INPUT_FIBER;
            end
        end

        if      area > VisionSide.FIBER_SIZE_THRESHOLD * VisionSide.WIDTH *
VisionSide.HEIGHT & ...
                aspect_ratio > VisionSide.FIBER_ASPECT_RATIO_THRESHOLD & ...
                rect(1) < VisionSide.FIBER_NEAR_EDGE_THRESHOLD *
VisionSide.WIDTH & ...
                rect(1) + rect(3) < (1 -
VisionSide.FIBER_NEAR_EDGE_THRESHOLD) * VisionSide.WIDTH

            output_fiber_index = i;
            if detected == Vision.DETECTED_INPUT_FIBER
```

```
                detected = Vision.DETECTED_TWO_FIBERS;
                break
            else
                detected = Vision.DETECTED_OUTPUT_FIBER;
            end

        end

        if      area > VisionSide.COUPLED_FIBERS_SIZE_THRESHOLD *
VisionSide.WIDTH * VisionSide.HEIGHT & ...
                aspect_ratio > 2 * VisionSide.FIBER_ASPECT_RATIO_THRESHOLD &
...
                rect(1) < VisionSide.FIBER_NEAR_EDGE_THRESHOLD *
VisionSide.WIDTH & ...
                rect(1) + rect(3) > (1 -
VisionSide.FIBER_NEAR_EDGE_THRESHOLD) * VisionSide.WIDTH

            coupled_fiber_index = i;
            detected = Vision.DETECTED_COUPLED_FIBERS;
            break
        end

    end

    if input_fiber_index >= 1
        rect = cv.boundingRect(contours{input_fiber_index});
        x = rect(1);
        y = rect(2);
        w = rect(3);
        h = rect(4);
        right_fiber_tip = thresh(y : y + h, x : x +
VisionSide.FIBER_TIP_PIXEL_WIDTH);
        contours_rft = cv.findContours(right_fiber_tip, 'Mode', 'List',
'Method', 'Simple');
        rect = cv.boundingRect(contours_rft{1});
        fiber_positions = [fiber_positions ; x, (y + rect(2) + rect(4) / 2)];
    end

    if output_fiber_index >=1
        rect = cv.boundingRect(contours{output_fiber_index});
        x = rect(1);
        y = rect(2);
        w = rect(3);
        h = rect(4);
        left_fiber_tip = thresh(y : y + h, x + w -
VisionSide.FIBER_TIP_PIXEL_WIDTH : x + w);
        contours_lft = cv.findContours(left_fiber_tip, 'Mode', 'List',
'Method', 'Simple');
        rect = cv.boundingRect(contours_lft{1});
        fiber_positions = [fiber_positions ; x + w, (y + rect(2) + rect(4) /
2)];
    end

    % Finding Pit
    template = imread('image-training/side image pit template.tif');
```

```
    if length(size(template)) == 3
        template = rgb2gray(template);
    end
    if is_input
        template = fliplr(template);
    end
    match = cv.matchTemplate(img, template, 'Method', 'CCorr');
    [rmax, cmax] = find(match==max(match(:)));
    if length(rmax) > 1
        rmax = rmax(1);
        cmax = cmax(1);
    end
    % Point 1 y, Point 1 x, Point 2 y, Point 2 x
    pit_position = [rmax, cmax, rmax + size(template, 1), cmax +
size(template, 2)];

end



end
```

Top Camera Vision Processing Algorithm:

```
function [ detected, fiber_positions, lateral_displacement, thresh ] =
process_top_image( img )
%{
% @param img the preprocessed side camera image
%
% @return detected What was detected in the image. Can be:
%   - DETECTED_UNKNOWN
%   - DETECTED_TWO_FIBERS
%   - DETECTED_COUPLED_FIBERS
%
% @return fiber_positions Array of [x, y] pairs describing the positions
%                         of the fibers (input, then output fiber). This is
only returned if
%                         two fibers are detected. Otherwise, an empty list
is returned
%
% @return lateral_displacement The float containing the lateral displacement
value, in pixels
%
% @return thresh The thresholded image for display purposes
%}

fiber_positions = [];
detected = Vision.DETECTED_UNKNOWN;
lateral_displacement = 0.0;


thresh_inv = cv.adaptiveThreshold(  img, 255, ...
```

```
                                    'AdaptiveMethod', 'Mean', ...
                                    'ThresholdType', 'Binary', ...
                                    'BlockSize',
VisionTop.ADAPTIVE_THRESH_KERNEL_SIZE, ...
                                    'C',
VisionTop.ADAPTIVE_THRESH_MEAN_MODIFIER);

thresh = cv.threshold(thresh_inv, 127, 'MaxValue', 255, 'Method',
'BinaryInv');

thresh = cv.morphologyEx(thresh, 'Open', 'Element',
VisionTop.OPENING_KERNEL_SIZE);
thresh = cv.morphologyEx(thresh, 'Close', 'Element',
VisionTop.CLOSING_KERNEL_SIZE);

contours = cv.findContours(thresh, 'Mode', 'List', 'Method', 'Simple');



if length(contours) > 0

    blobs = cell(length(contours), 2);
    for i=1:length(contours)
        blobs(i, :) = {contours{i}, cv.contourArea(contours{i})};
    end
    blobs = flipud(sortrows(blobs, 2));
    contours = blobs(:, 1)';

    input_fiber_index = -1;
    output_fiber_index = -1;
    for i=1:length(contours)
        hull = cv.convexHull(contours{i});
        area = cv.contourArea(hull);
        rect = cv.minAreaRect(hull);
        aspect_ratio = max(rect.size(1) / rect.size(2), rect.size(2) /
rect.size(1));

        if      area > VisionTop.FIBER_SIZE_THRESHOLD * VisionTop.WIDTH *
VisionTop.HEIGHT & ...
                aspect_ratio > VisionTop.FIBER_ASPECT_RATIO_THRESHOLD & ...
                rect.center(2) > VisionTop.HEIGHT / 2 & ...
                i ~= output_fiber_index

            input_fiber_index = i;
            if detected == Vision.DETECTED_OUTPUT_FIBER
                detected = Vision.DETECTED_TWO_FIBERS;
                break
            else
                detected = Vision.DETECTED_INPUT_FIBER;
            end

        end
```

```
        if      area > VisionTop.FIBER_SIZE_THRESHOLD * VisionTop.WIDTH *
VisionTop.HEIGHT & ...
                aspect_ratio > VisionTop.FIBER_ASPECT_RATIO_THRESHOLD & ...
                rect.center(2) <= VisionTop.HEIGHT / 2 & ...
                i ~= input_fiber_index

            output_fiber_index = i;
            if detected == Vision.DETECTED_INPUT_FIBER
                detected = Vision.DETECTED_TWO_FIBERS;
                break
            else
                detected = Vision.DETECTED_OUTPUT_FIBER;
            end

        end

    end

    if input_fiber_index >=1
        input_line = cv.fitLine(cv.convexHull(contours{input_fiber_index}));
        fiber_positions = [fiber_positions ; input_line'];
    end

    if output_fiber_index >= 1
        output_line =
cv.fitLine(cv.convexHull(contours{output_fiber_index}));
        fiber_positions = [fiber_positions ; output_line'];
    end

    if detected == Vision.DETECTED_TWO_FIBERS
        lateral_displacement = get_lateral_displacement(input_line,
output_line);
    end

end



end
```

Camera Initialization Code:

```
% --- Executes on button press in button_initialize_cameras.
function button_initialize_cameras_Callback(hObject, eventdata, handles)
% hObject    handle to button_initialize_cameras (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global cap_top cap_side;
if strcmp(get(handles.text_display_cameras, 'String'), 'Disabled')
    disp('Initializing Cameras...')
    cap_top = cv.VideoCapture(0);
    pause(3);
```

```
    cap_side = cv.VideoCapture(1);
    pause(3);
    cap_top.set('FrameWidth', VisionTop.RAW_WIDTH);
    cap_top.set('FrameHeight', VisionTop.RAW_HEIGHT);
    cap_side.set('FrameWidth', VisionSide.RAW_WIDTH);
    cap_side.set('FrameHeight', VisionSide.RAW_HEIGHT);
    while cap_top.get('FrameWidth') ~= VisionTop.RAW_WIDTH
        disp(cap_top.get('FrameWidth'))
    end
    while cap_side.get('FrameWidth') ~= VisionSide.RAW_WIDTH
        disp(cap_side.get('FrameWidth'))
    end

    set(handles.text_display_cameras, 'String', 'Enabled');

end
```

Piezoelectric Actuator Initialization Code:

```
% --- Executes on button press in button_initialize_piezos.
function button_initialize_piezos_Callback(hObject, eventdata, handles)
% hObject    handle to button_initialize_piezos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global h_Ctrl output_x_ctrl output_y_ctrl output_z_ctrl output_yaw_ctrl
input_x_ctrl input_y_ctrl input_z_ctrl fig
if strcmp(get(handles.text_display_piezos, 'String'), 'Disabled')
    fig = figure;
    %fig.Visible = 'off';

    h_Ctrl = actxcontrol('MG17SYSTEM.MG17SystemCtrl.1', [0 0 100 100], fig);
    h_Ctrl.StartCtrl;

    output_x_ctrl = actxcontrol('MGPIEZO.MGPiezoCtrl.1', [0 200 300 200],
fig);
    output_x_ctrl.StartCtrl;

    output_y_ctrl = actxcontrol('MGPIEZO.MGPiezoCtrl.1', [0 200 300 200],
fig);
    output_y_ctrl.StartCtrl;

    output_z_ctrl = actxcontrol('MGPIEZO.MGPiezoCtrl.1', [0 200 300 200],
fig);
    output_z_ctrl.StartCtrl;

    input_x_ctrl = actxcontrol('MGPIEZO.MGPiezoCtrl.1', [0 200 300 200],
fig);
    input_x_ctrl.StartCtrl;

    input_y_ctrl = actxcontrol('MGPIEZO.MGPiezoCtrl.1', [0 200 300 200],
fig);
    input_y_ctrl.StartCtrl;
```

```
    input_z_ctrl = actxcontrol('MGPIEZO.MGPiezoCtrl.1', [0 200 300 200],
fig);
    input_z_ctrl.StartCtrl;

    jog_step_size = 0.25;

    set(output_x_ctrl, 'HWSerialNum', 91815963);
    pause(0.1)
    set(output_y_ctrl, 'HWSerialNum', 91815962);
    pause(0.1)
    set(output_z_ctrl, 'HWSerialNum', 91815961);
    pause(0.1)

    set(input_x_ctrl, 'HWSerialNum', 91815973);
    pause(0.1)
    set(input_y_ctrl, 'HWSerialNum', 91815972);
    pause(0.1)
    set(input_z_ctrl, 'HWSerialNum', 91815971);
    pause(0.1)

    output_x_ctrl.SetControlMode(0, 4);
    output_x_ctrl.SetVoltPosDispMode(0, 2);
    output_y_ctrl.SetControlMode(0, 4);
    output_y_ctrl.SetVoltPosDispMode(0, 2);
    output_z_ctrl.SetControlMode(0, 4);
    output_z_ctrl.SetVoltPosDispMode(0, 2);

    input_x_ctrl.SetControlMode(0, 4);
    input_x_ctrl.SetVoltPosDispMode(0, 2);
    input_y_ctrl.SetControlMode(0, 4);
    input_y_ctrl.SetVoltPosDispMode(0, 2);
    input_z_ctrl.SetControlMode(0, 4);
    input_z_ctrl.SetVoltPosDispMode(0, 2);

    ret = output_x_ctrl.SetPIConsts(0, 100, 100); % P=100, I=100 for x-axis
    ret = output_y_ctrl.SetPIConsts(0, 100, 95); % P=100, I=95
    ret = output_z_ctrl.SetPIConsts(0, 100, 95); % P=100, I=95

    ret = input_x_ctrl.SetPIConsts(0, 100, 100); % P=100, I=100 for x-axis
    ret = input_y_ctrl.SetPIConsts(0, 100, 95); % P=100, I=95
    ret = input_z_ctrl.SetPIConsts(0, 100, 95); % P=100, I=95

    output_x_ctrl.SetPosOutput(0, 0.0   );
    output_y_ctrl.SetPosOutput(0, 15.0);
    output_z_ctrl.SetPosOutput(0, 15.0);

    input_x_ctrl.SetPosOutput(0, 0.0   );
    input_y_ctrl.SetPosOutput(0, 15.0);
    input_z_ctrl.SetPosOutput(0, 15.0);

    pause(0.5); % to let oscillations die down a bit;
    set(handles.text_display_piezos, 'String', 'Enabled');
end
```

Power Meter Initialization Code:

```
% --- Executes on button press in button_initialize_power_meter.
function button_initialize_power_meter_Callback(hObject, eventdata, handles)
% hObject    handle to button_initialize_power_meter (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global power_meter;
if strcmp(get(handles.text_display_power_meter, 'String'), 'Disabled')
    power_meter = instrfind('Type', 'gpib', 'BoardIndex', 0,
'PrimaryAddress', 5, 'Tag', '');
    % Create the GPIB object if it does not exist
    % otherwise use the object that was found.
    if isempty(power_meter)
        power_meter = gpib('NI', 0, 5);
    else
        fclose(power_meter);
        power_meter = power_meter(1);
    end
    fopen(power_meter);
    set(handles.text_display_power_meter, 'String', 'Enabled');
    drawnow;
end
```

Camera Release Code:

```
% --- Executes on button press in button_release_cameras.
function button_release_cameras_Callback(hObject, eventdata, handles)
% hObject    handle to button_release_cameras (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global cap_side cap_top;
if strcmp(get(handles.text_display_cameras, 'String'), 'Enabled')
    cap_side.release();
    cap_top.release();
    clear cap_side;
    clear cap_top;
    set(handles.text_display_cameras, 'String', 'Disabled');
end
```

Piezoelectric Actuator Release Code:

```
% --- Executes on button press in button_release_piezos.
function button_release_piezos_Callback(hObject, eventdata, handles)
% hObject    handle to button_release_piezos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global fig output_x_ctrl output_y_ctrl output_z_ctrl output_yaw_ctrl
input_x_ctrl input_y_ctrl input_z_ctrl h_Ctrl;
if strcmp(get(handles.text_display_piezos, 'String'), 'Enabled')
    output_x_ctrl.StopCtrl;
    output_y_ctrl.StopCtrl;
    output_z_ctrl.StopCtrl;
```

```
    input_x_ctrl.StopCtrl;
    input_y_ctrl.StopCtrl;
    input_z_ctrl.StopCtrl;
    %output_yaw_ctrl.StopCtrl;
    h_Ctrl.StopCtrl;
    close(fig);
    set(handles.text_display_piezos, 'String', 'Disabled');
end
```

Power Meter Release Code:

```
% --- Executes on button press in button_release_power_meter.
function button_release_power_meter_Callback(hObject, eventdata, handles)
% hObject    handle to button_release_power_meter (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global power_meter;
if ~strcmp(get(handles.text_display_power_meter), 'Disabled')
    fclose(power_meter);
    set(handles.text_display_power_meter, 'String', 'Disabled');
else
    % debug case where we don't actually have a physical power meter to
    % close
    set(handles.text_display_power_meter, 'String', 'Disabled');
end
```